BERLIN
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2013

goto; conference

Training:Oct 15-16 / Conference:Oct 17-18

managing micro-services

distributed systems and your infrastructure

jalewis@thoughtworks.com : @boicy : http://bovon.org

ThoughtWorks®

todays topics

recap on micro-services

some approaches to building them

some approaches to deploying them

Technology Radar

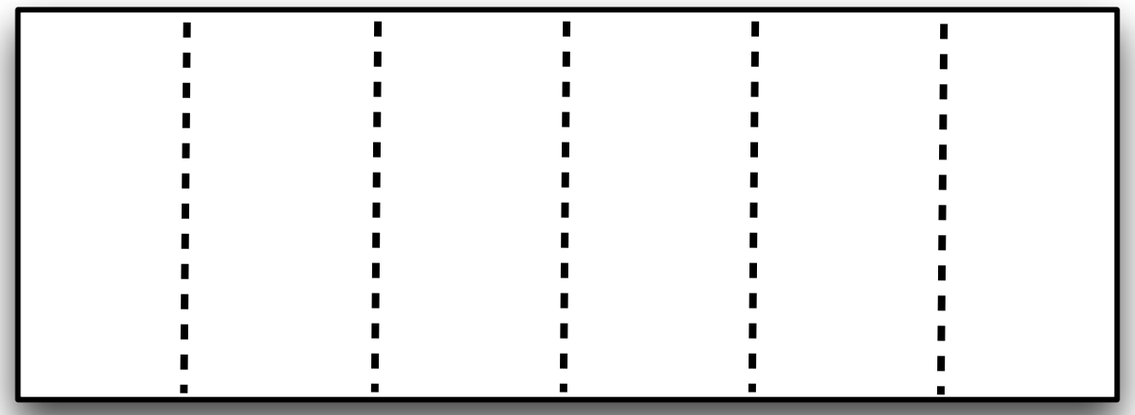
Prepared by the ThoughtWorks Technology Advisory Board

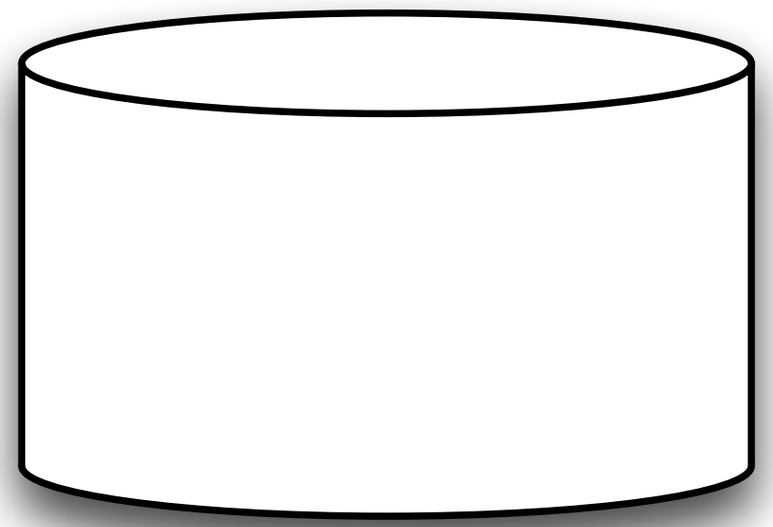
OCTOBER 2012

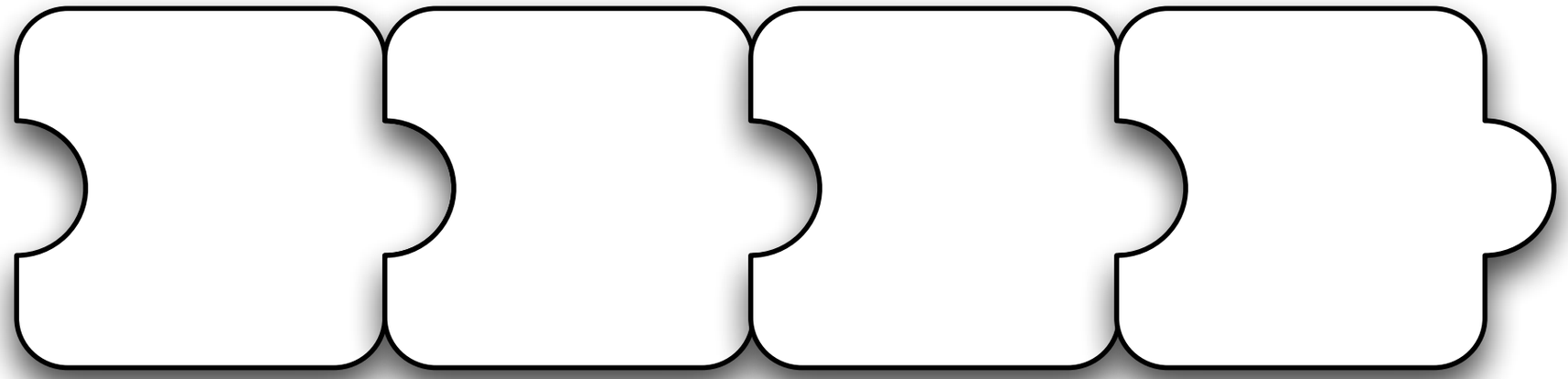
thoughtworks.com/radar

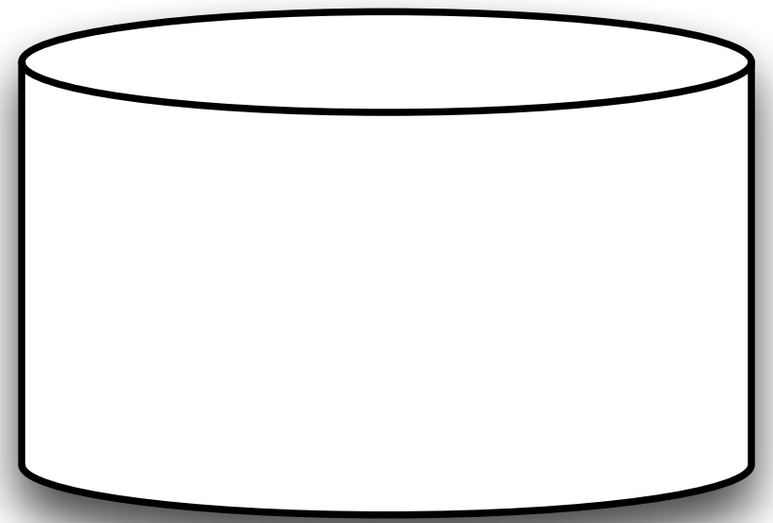
the story so far

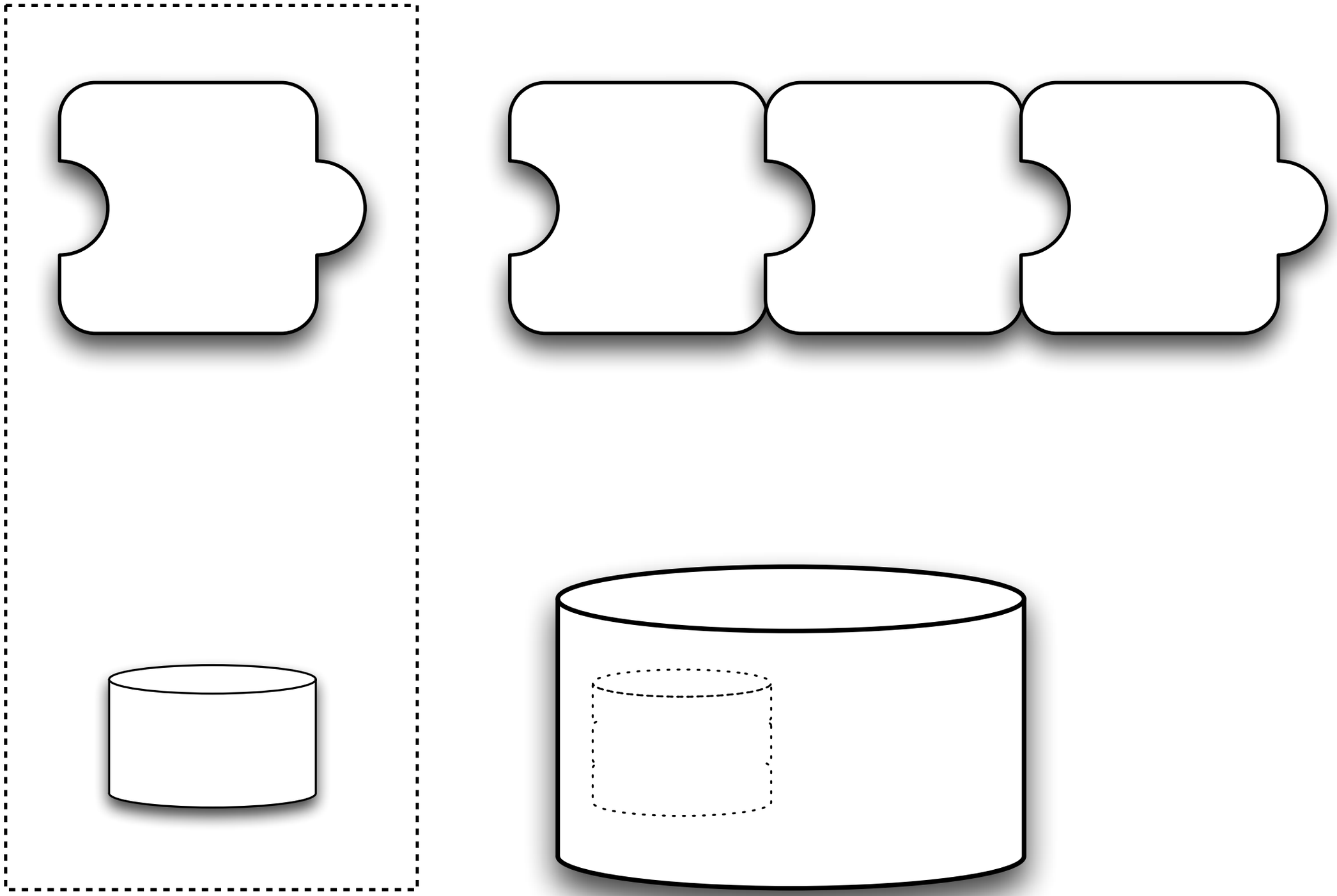
Users Rules Reports Analytics



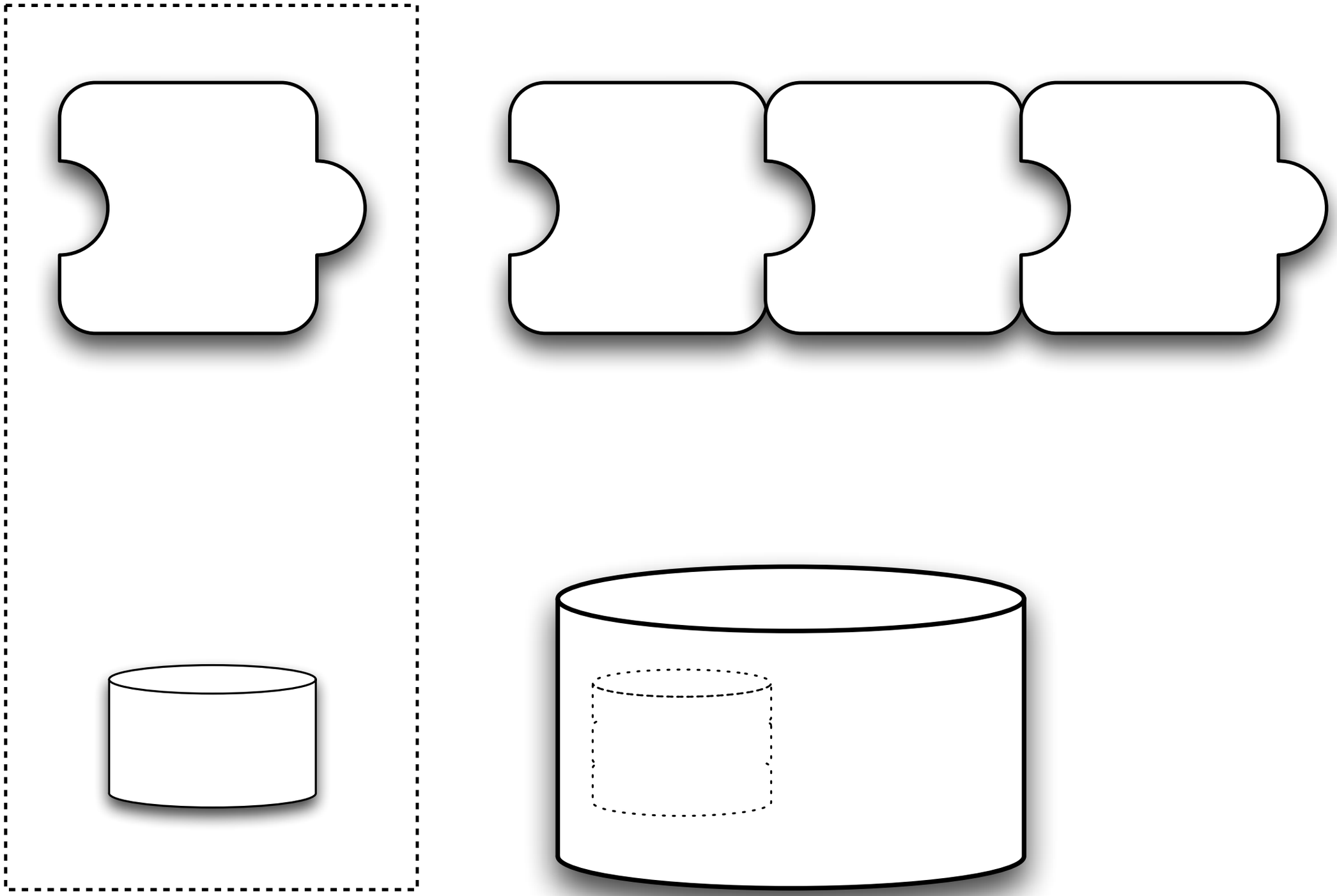


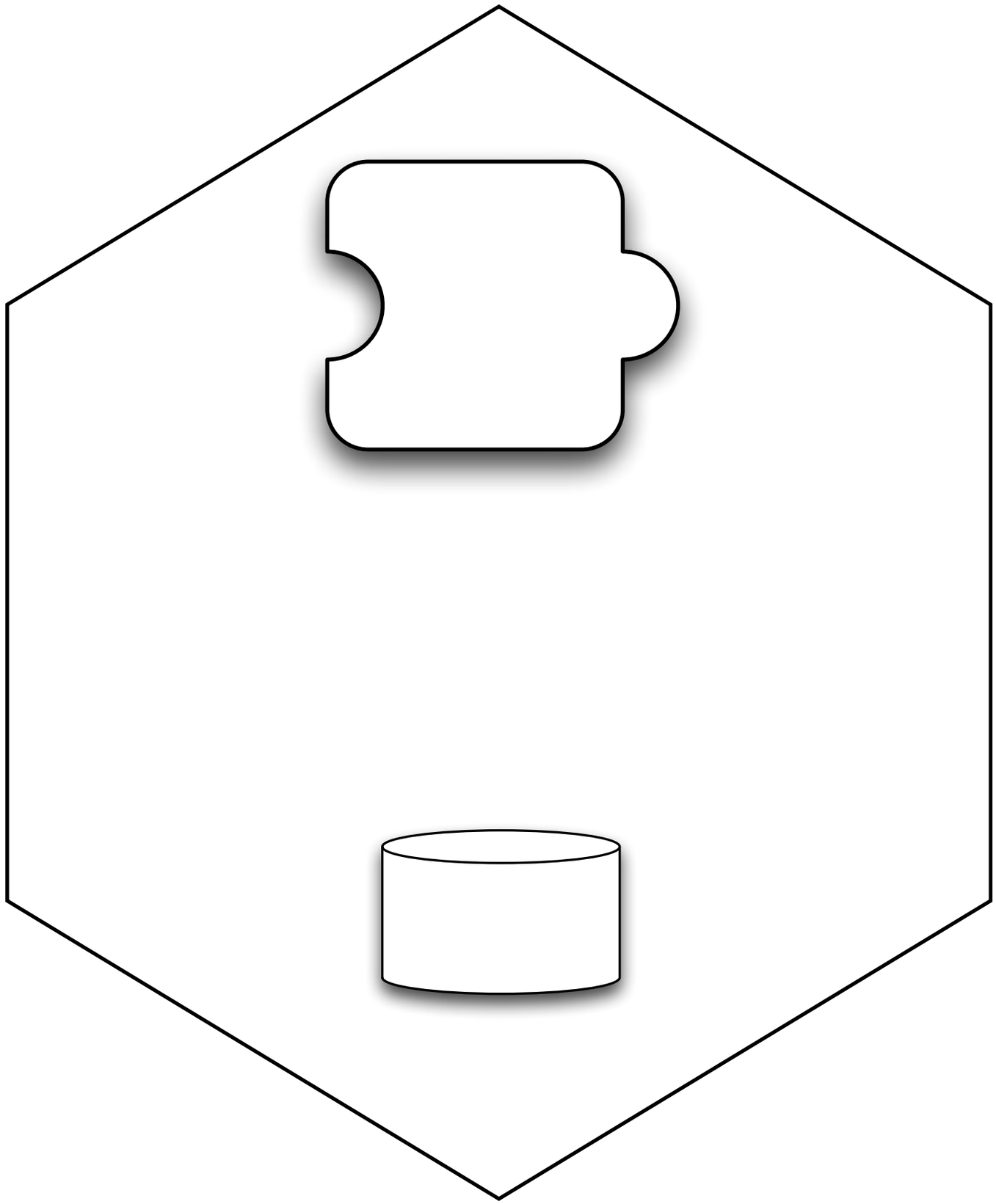


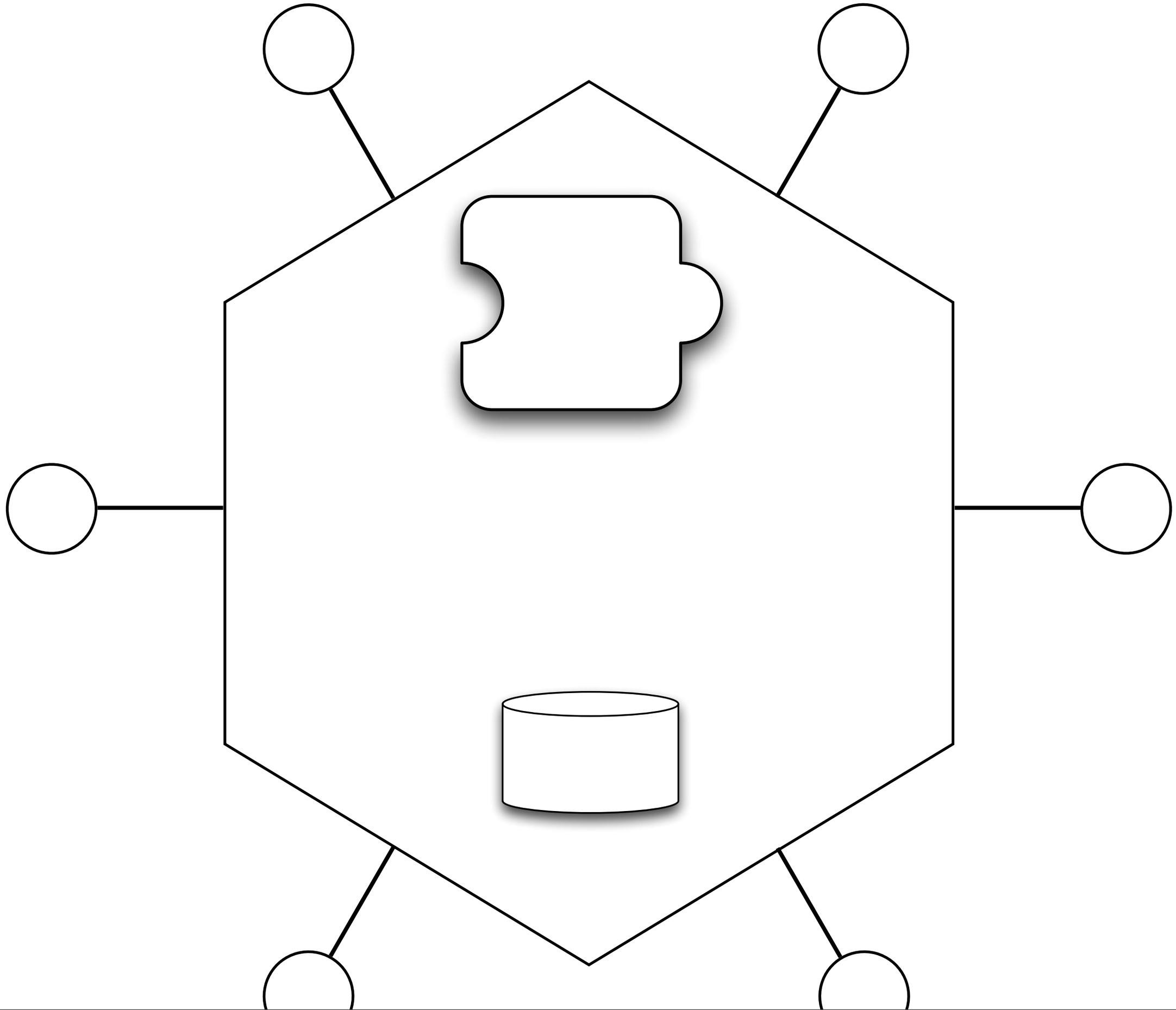


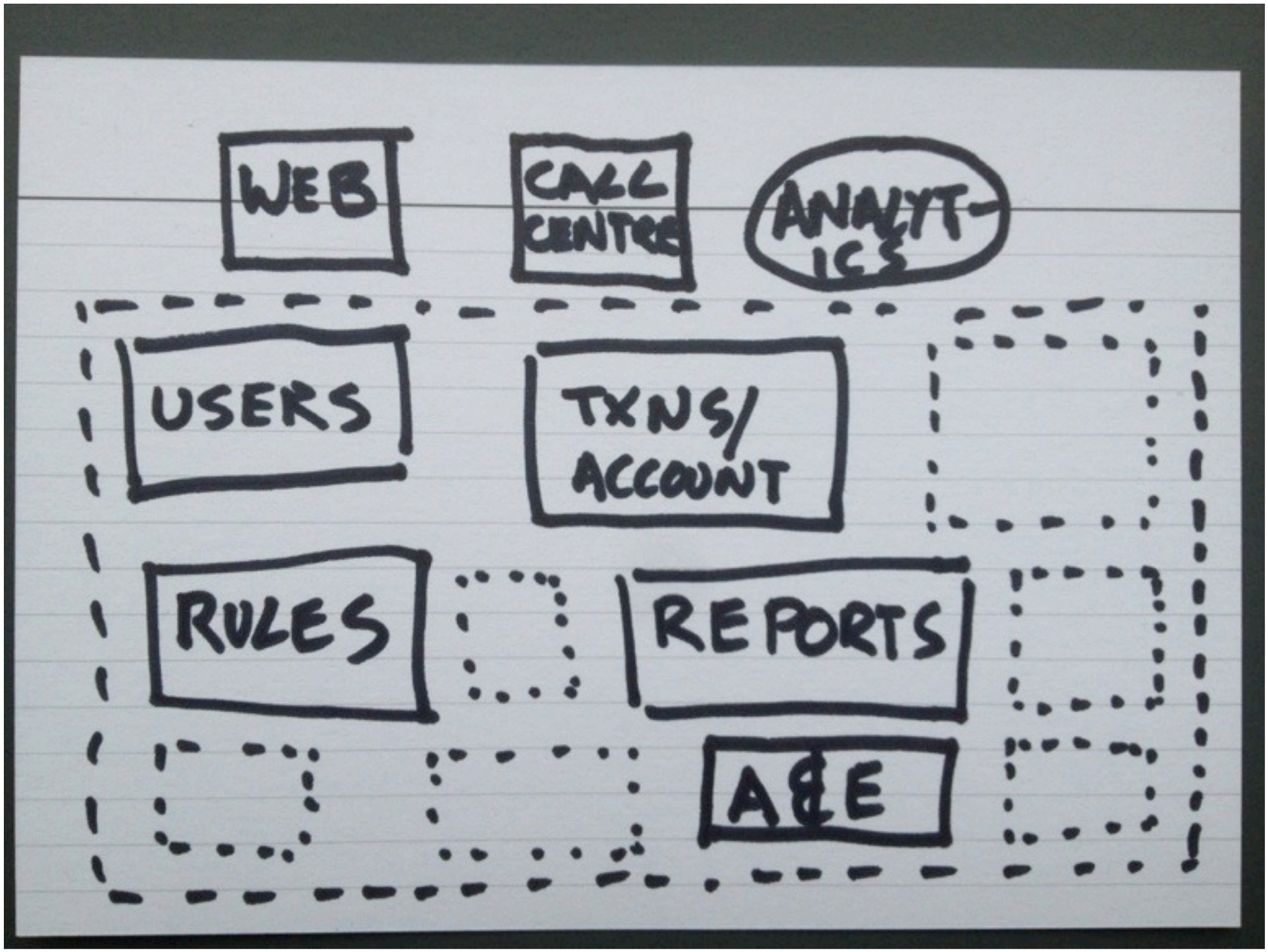


hexagonal business capabilities
owning their own data

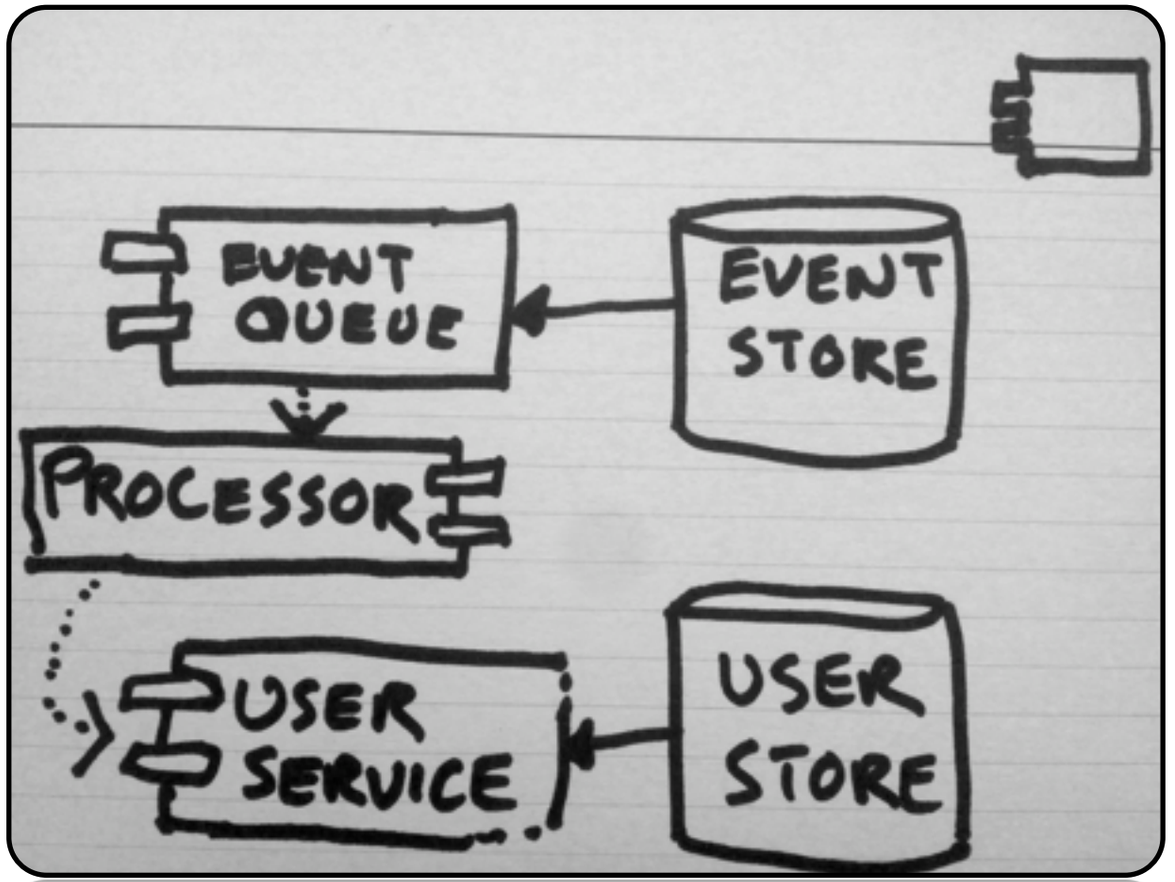


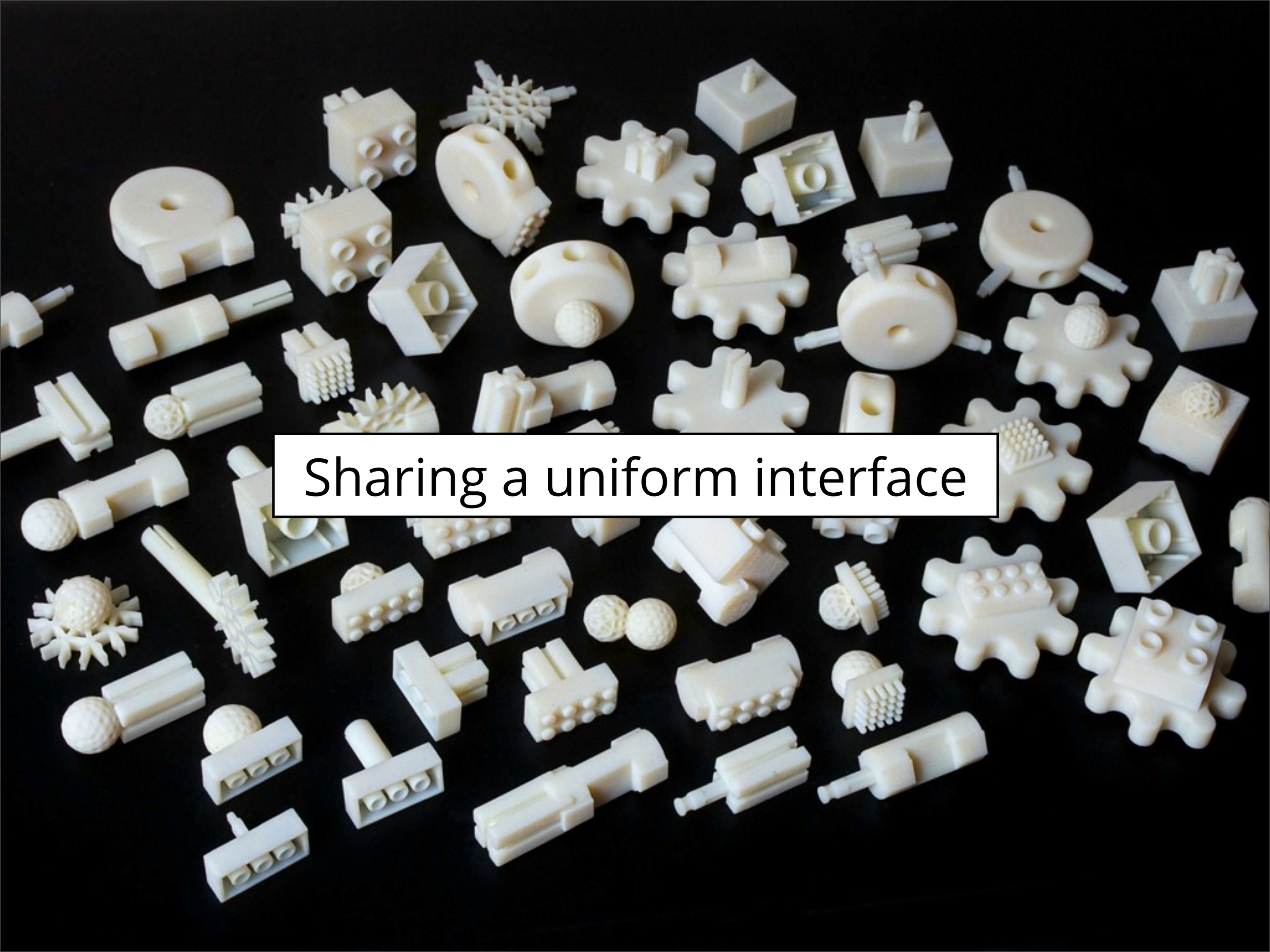




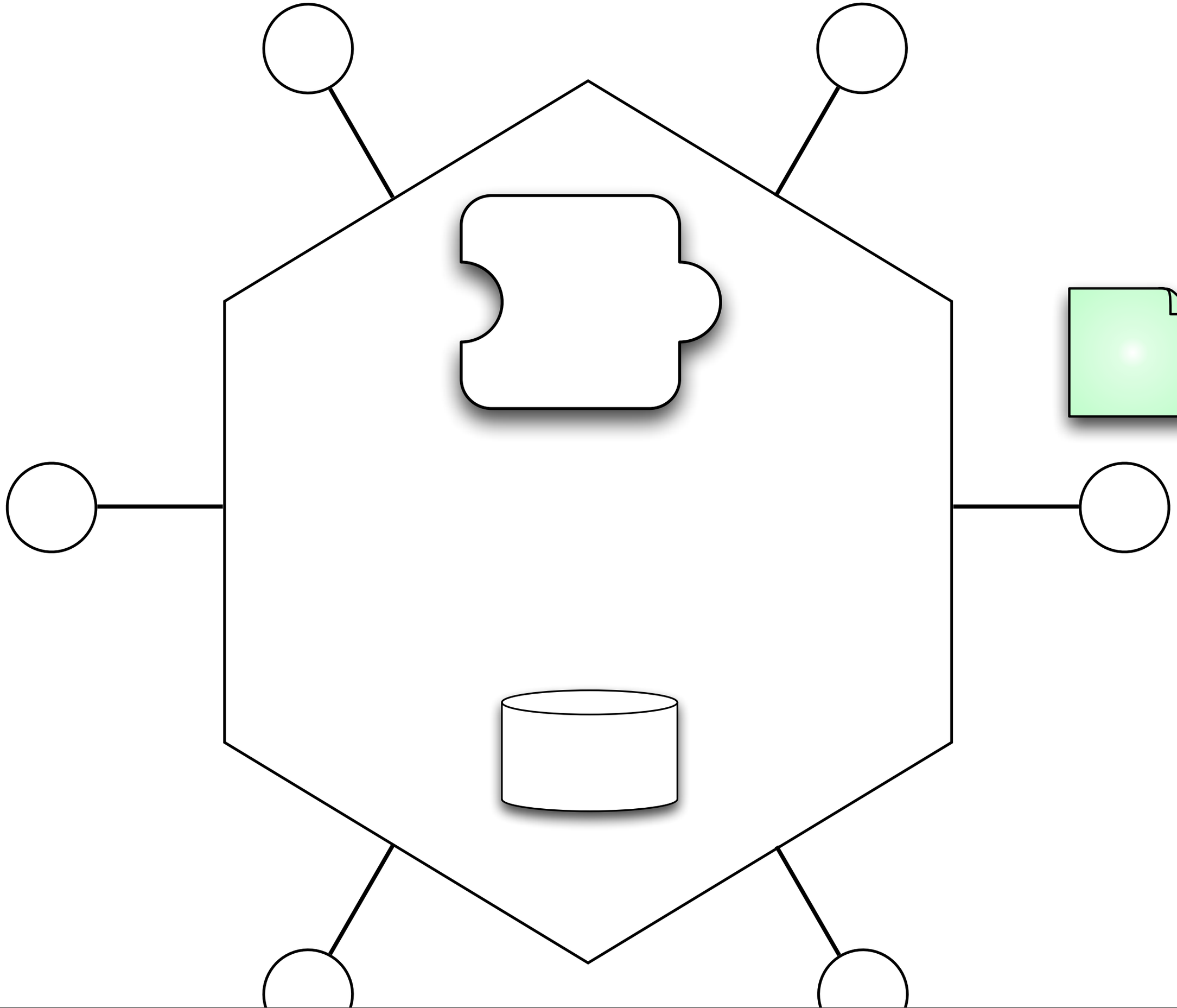

WEB
CALL CENTRE
ANALYT-ICS
USERS
TXNS/ ACCOUNT
RULES
REPORTS
A&E

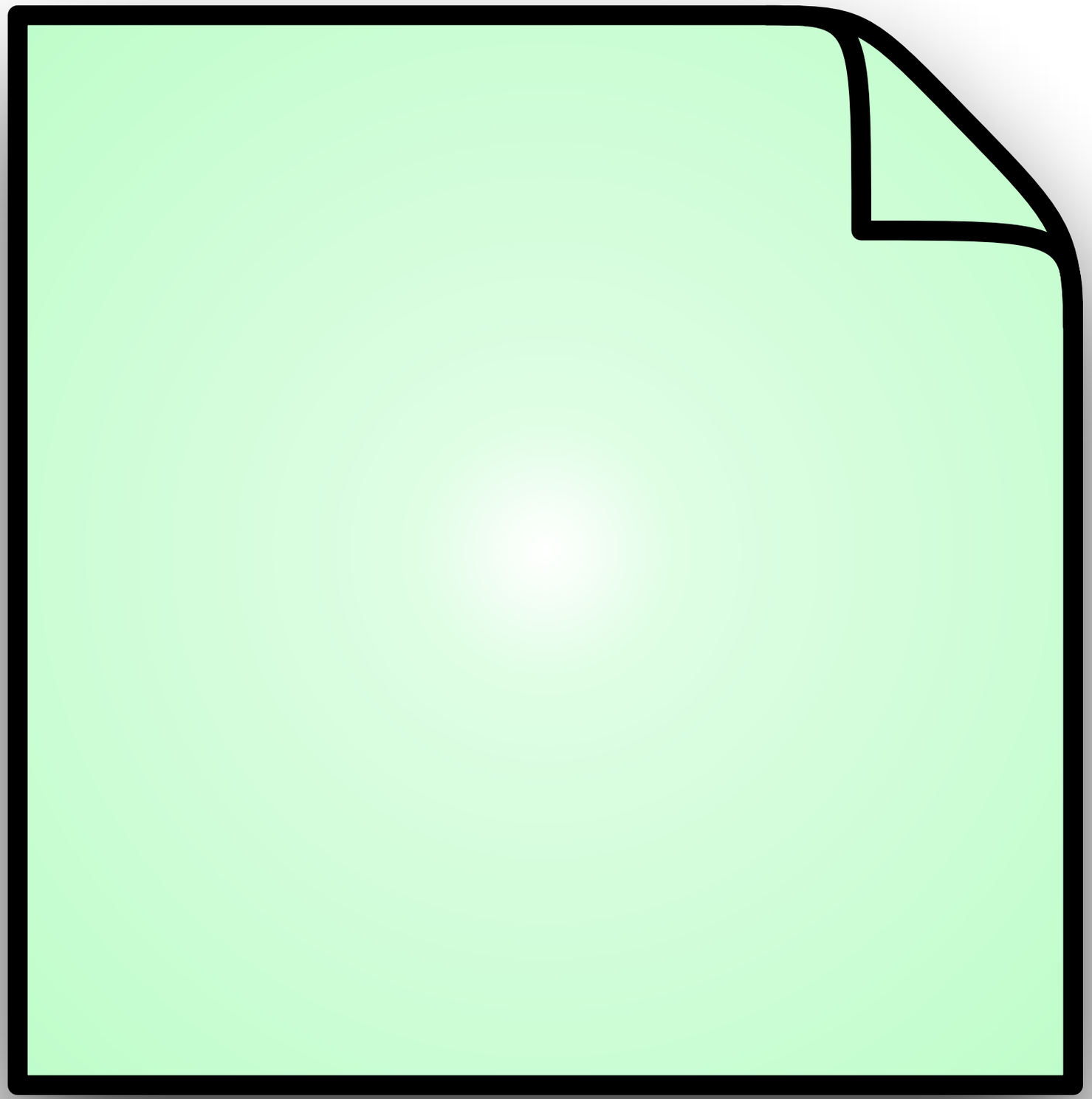
Each capability decomposed into smaller applications based on your **functional** and **cross-functional** requirements



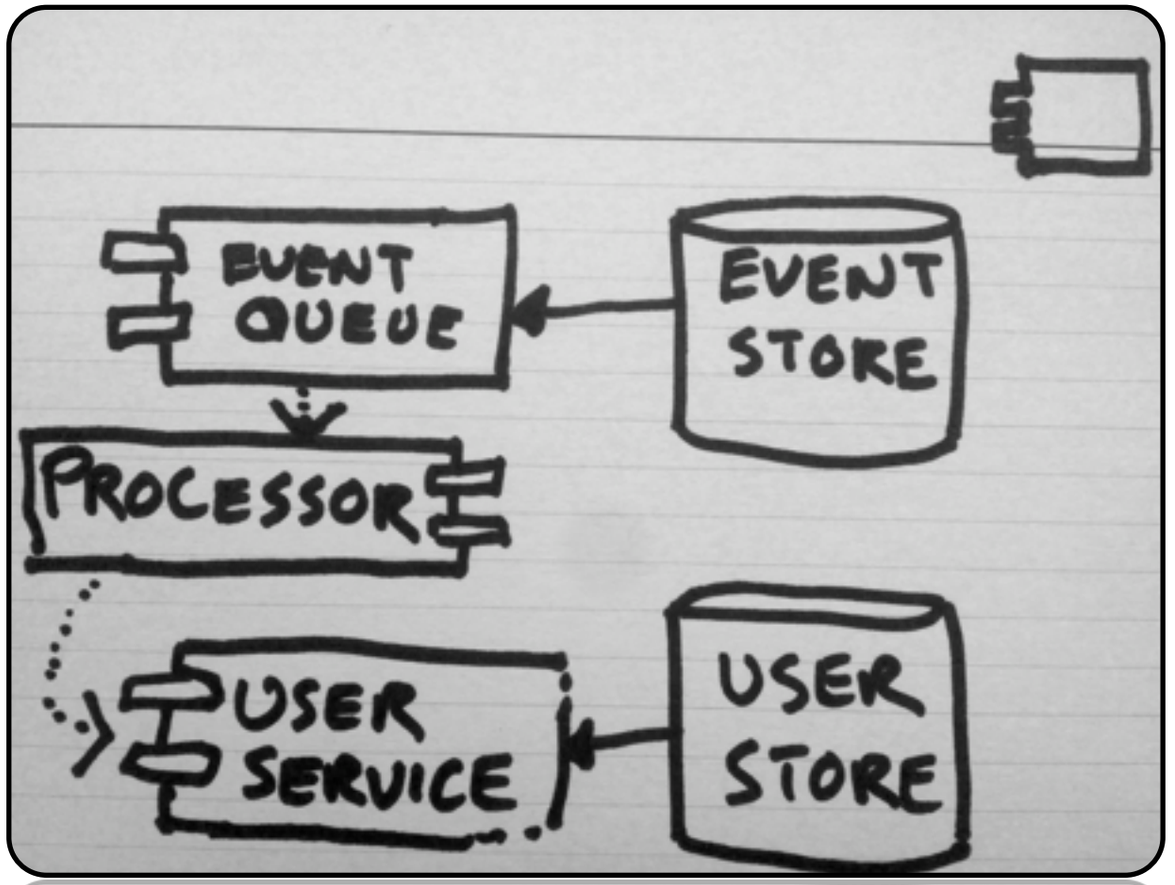


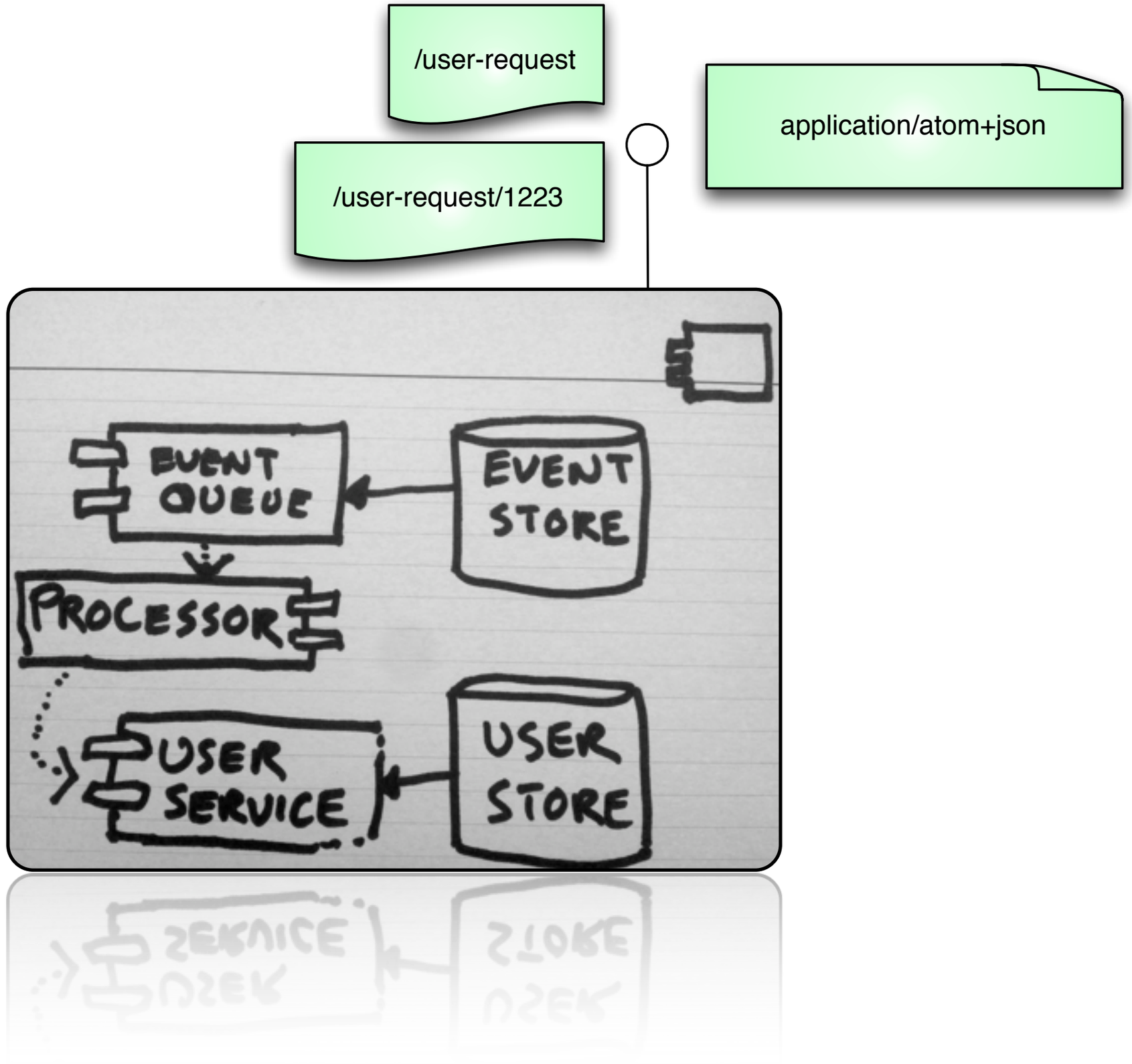
Sharing a uniform interface





```
<html>
<body>
      <div>
          <span id="firstName">James</span>
          <span id="lastName">Lewis</span>
      </div>
      <div id="links">
          <a rel="self"
             href="/users/ae3fc"
             type="application/vnd.foobar.user+xml">Register User</a>

          <form id="update" method="POST" name="updateForm" action="/users/ae3fc">
              <input id="updatedFirstName" type="text" name="firstName" value="James"/>
              <input type="text" name="lastName" value="lewis"/>
              <input type="submit" name="updateButton" value="update"/>
          </form>

          <a rel="userIndex"
             href="/users/index"
             type="text/html">Search Users</a>>
      </div>
  </body>
</html>
```

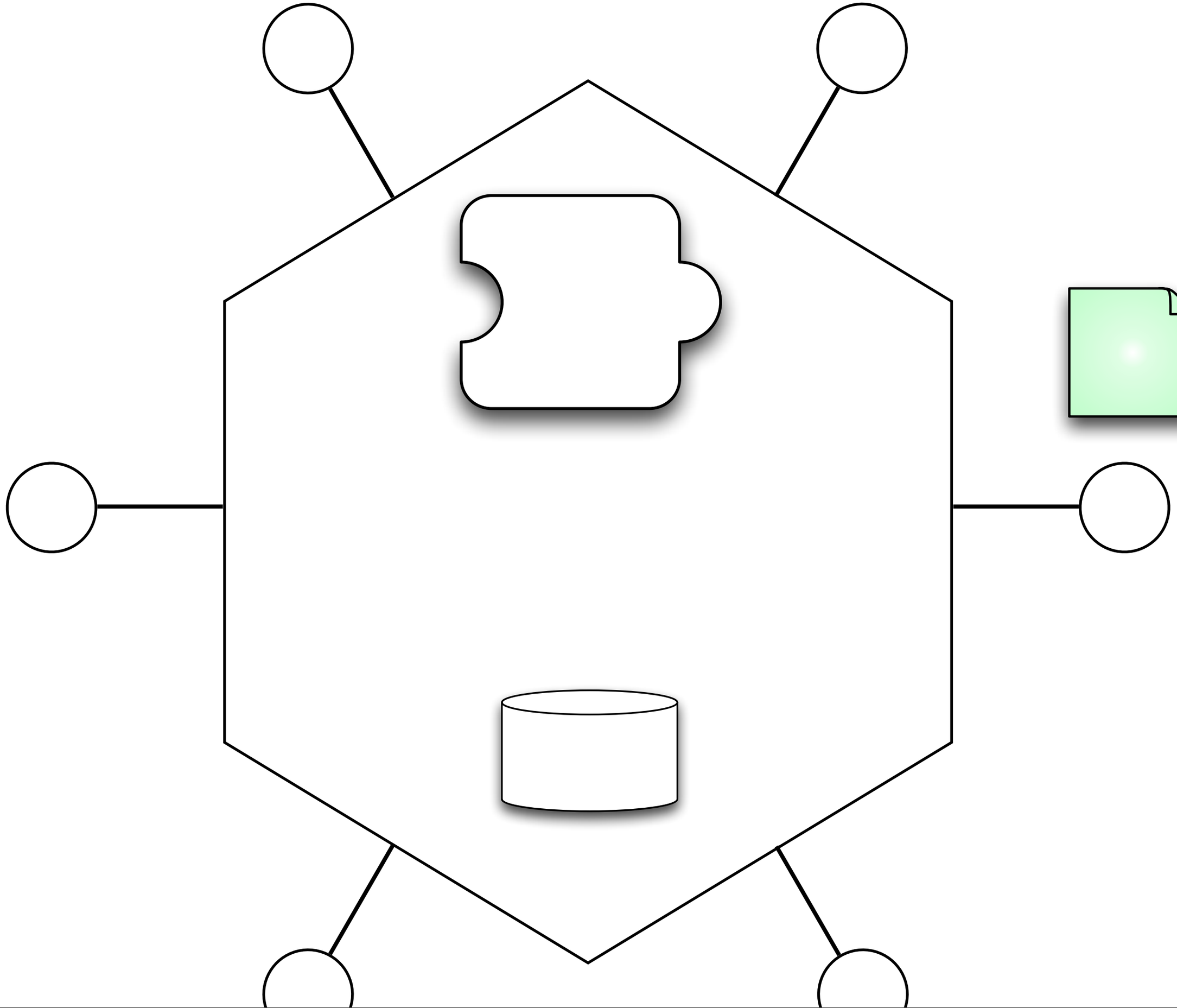

EVENT QUEUE
EVENT STORE
PROCESSOR
USER SERVICE
USER STORE

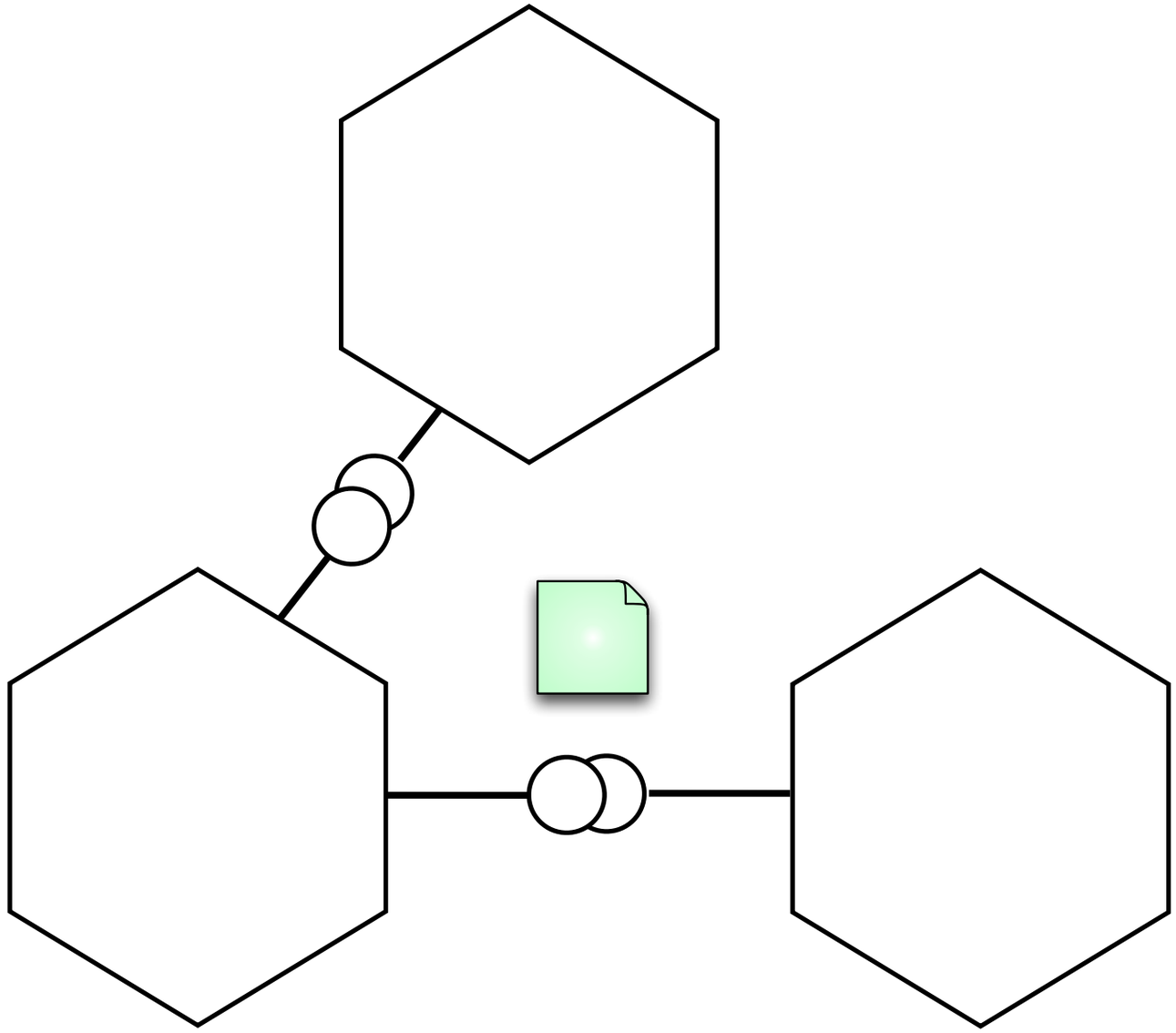

/user-request
/user-request/1223
application/atom+json
EVENT QUEUE
EVENT STORE
PROCESSOR
USER SERVICE
USER STORE

Use standard application protocols to bridge the semantic gap

TIME FLIES LIKE AN ARROW

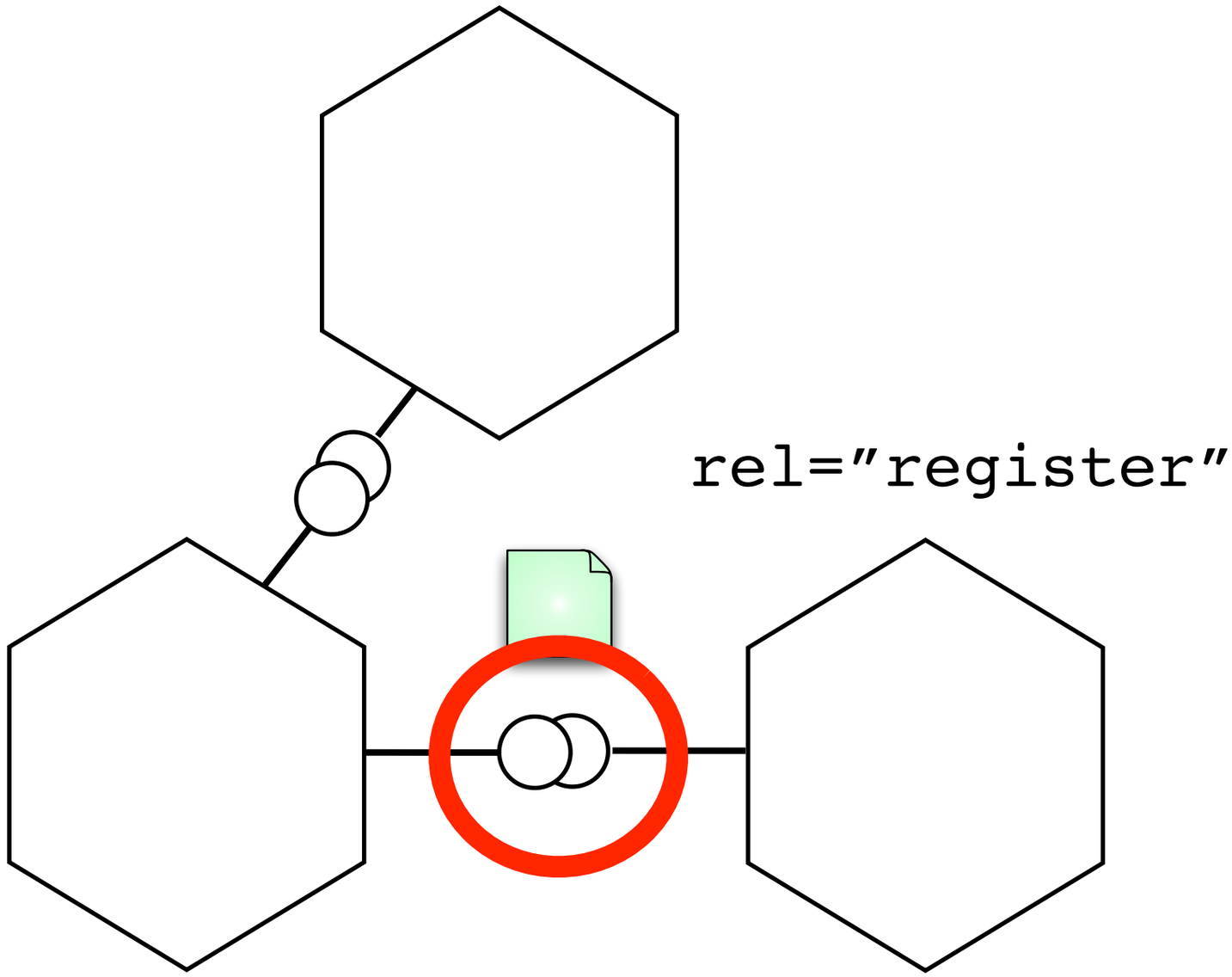
TIME LORDS LIKE A TARDIS





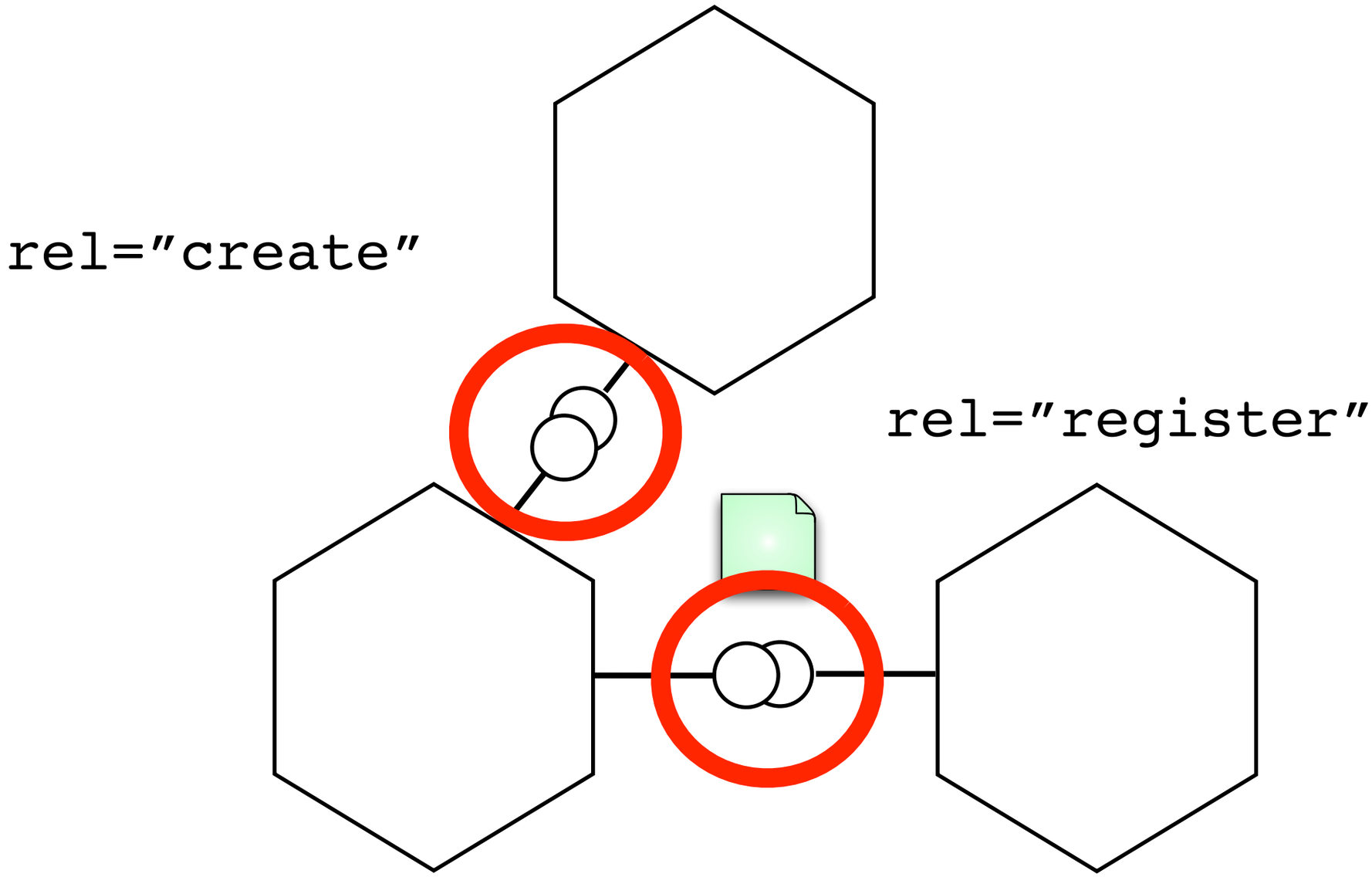
The **semantic gap** characterises the difference between two descriptions of an object by different linguistic representations, for instance languages or symbols.

http://en.wikipedia.org/wiki/Semantic_gap



The semantic gap characterises the difference between two descriptions of an object by different linguistic representations, for instance languages or symbols.

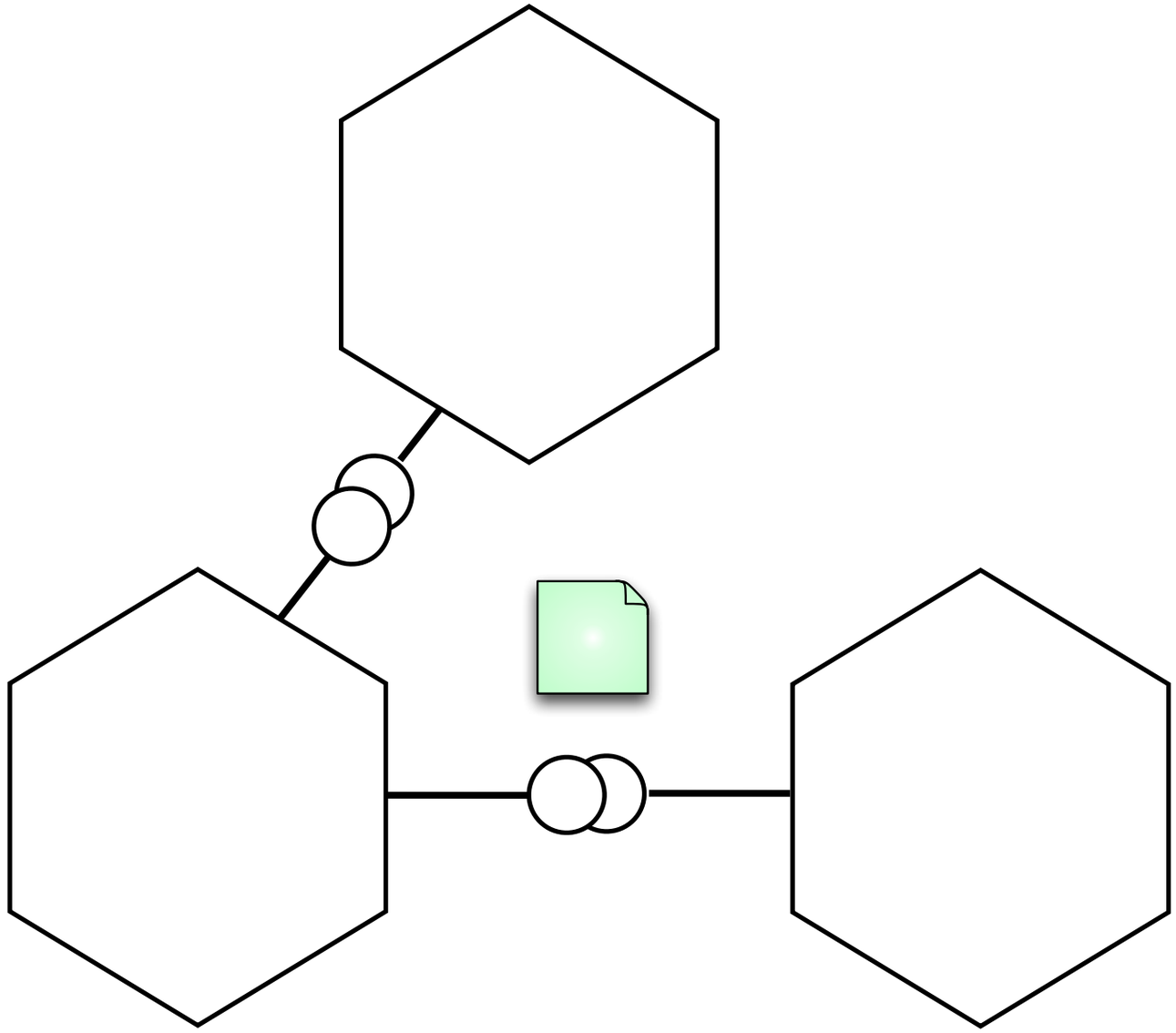
http://en.wikipedia.org/wiki/Semantic_gap

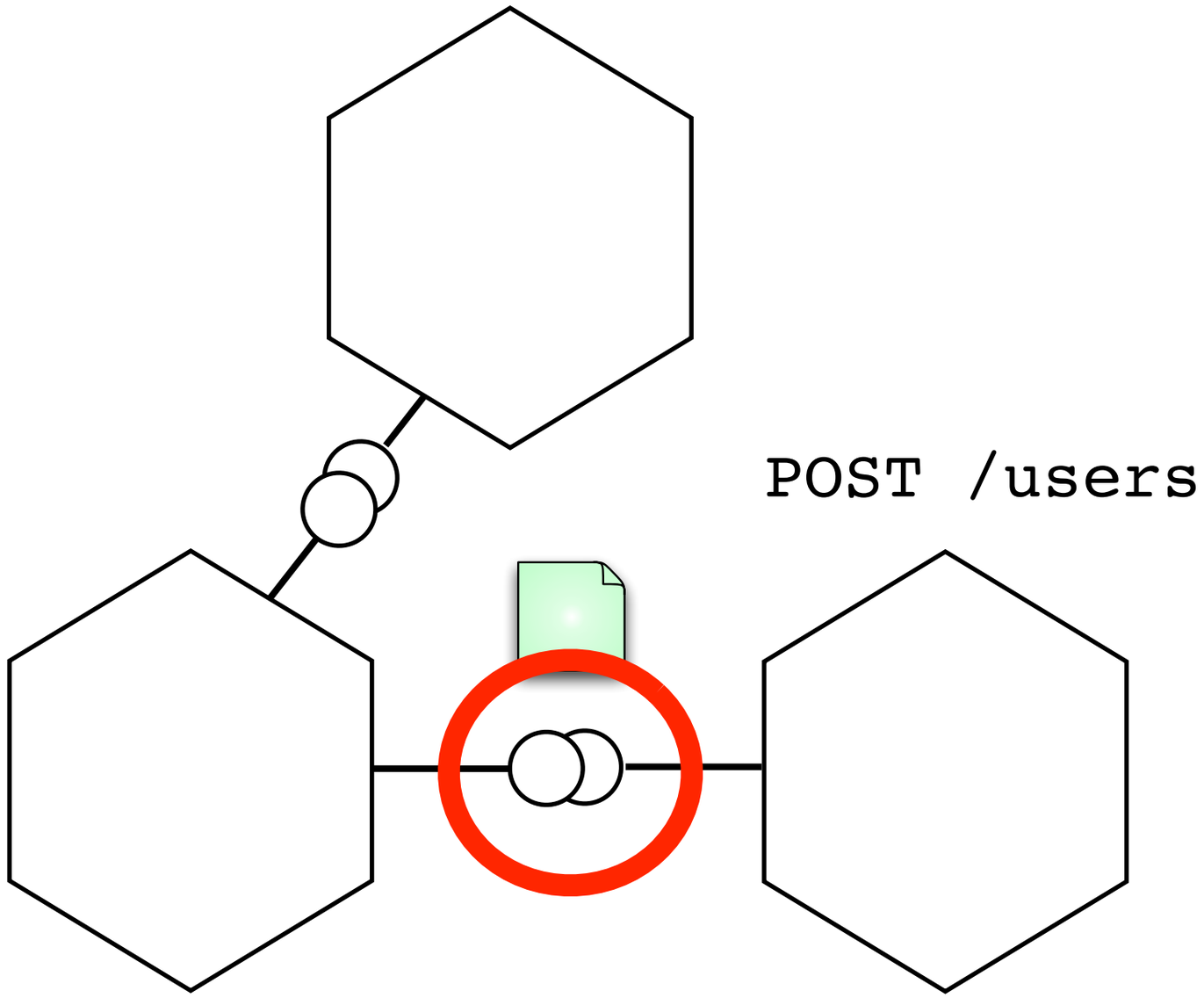


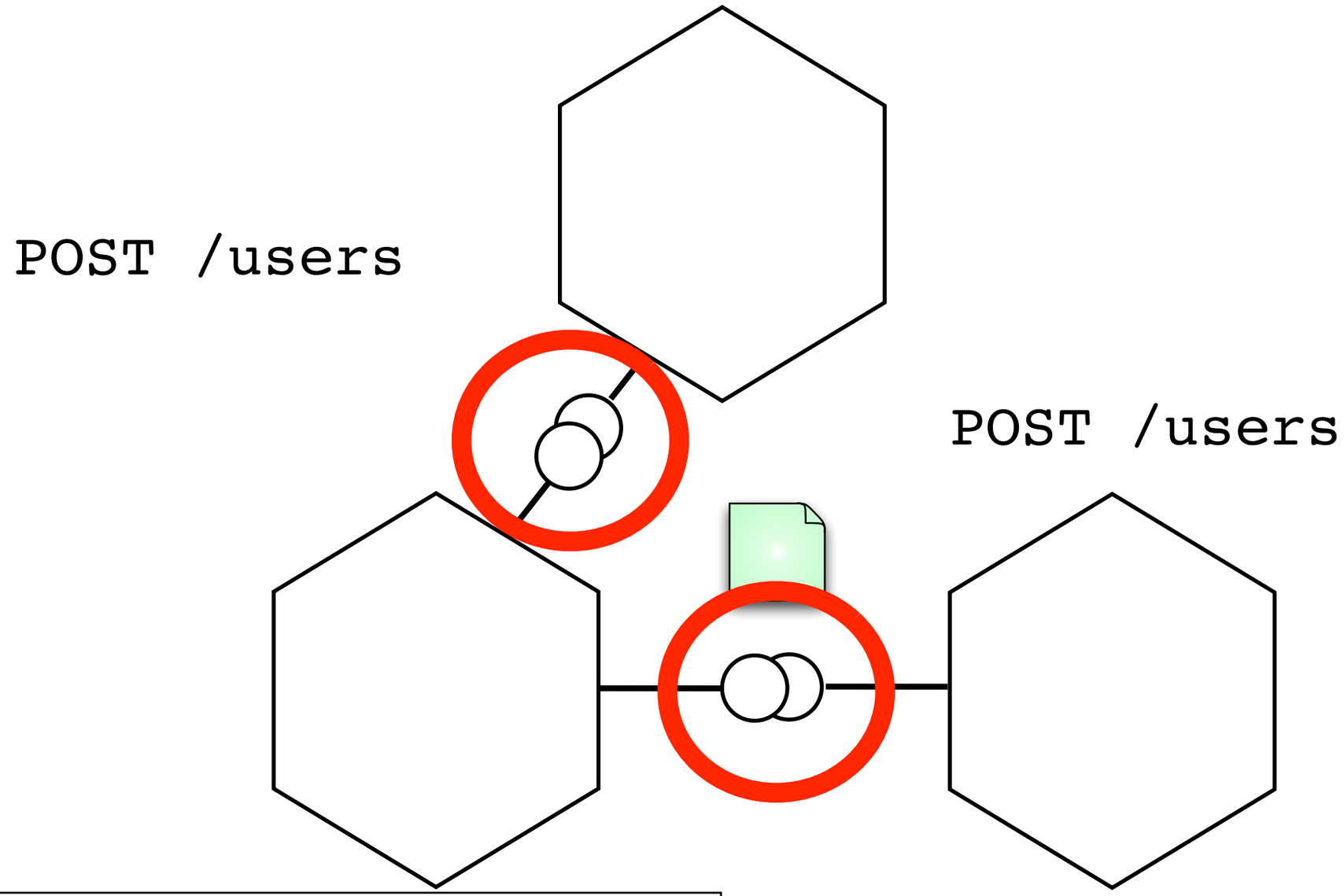
The **semantic gap** characterises the difference between two descriptions of an object by different linguistic representations, for instance languages or symbols.

http://en.wikipedia.org/wiki/Semantic_gap

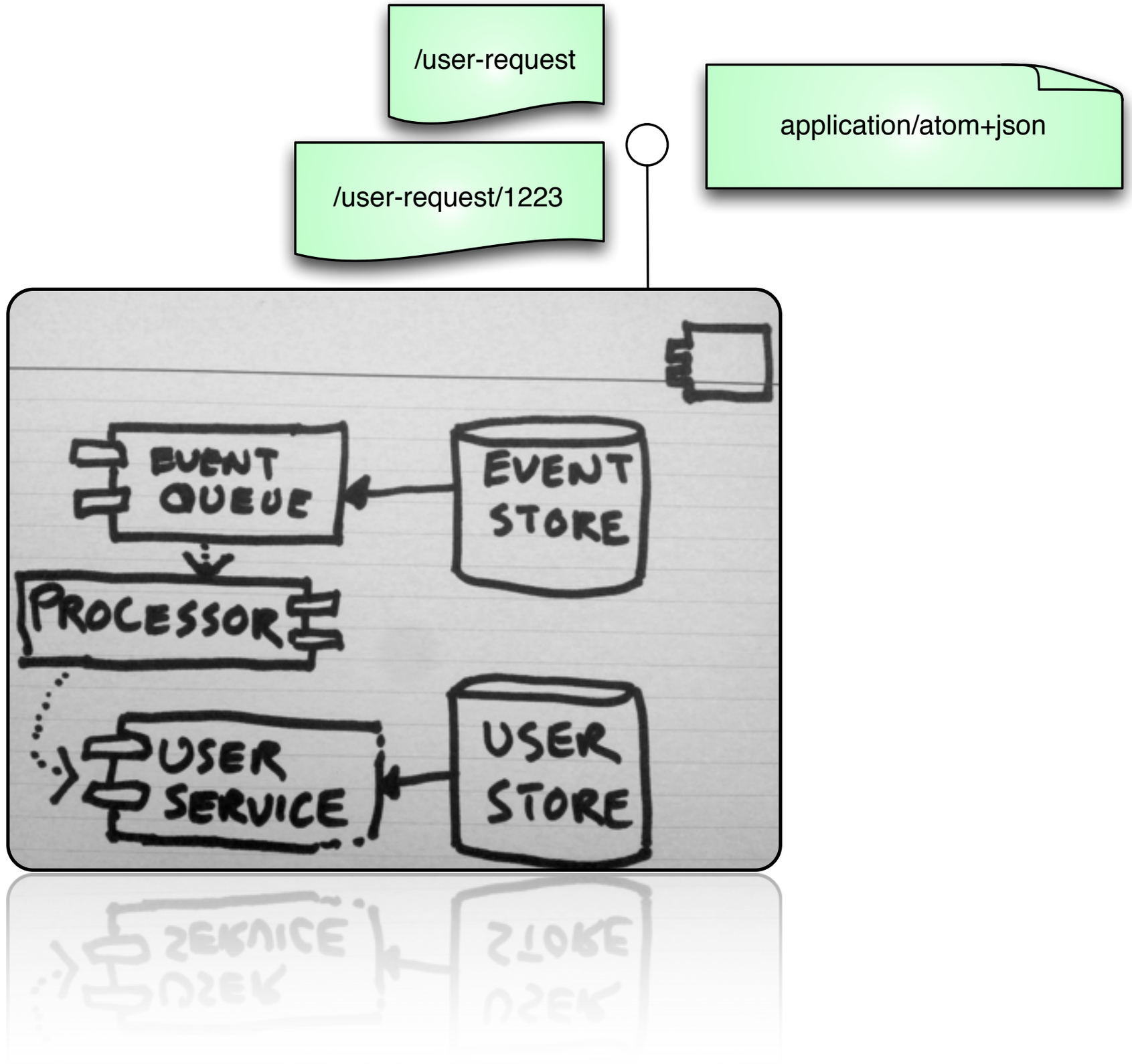
RFC 5023 is an example



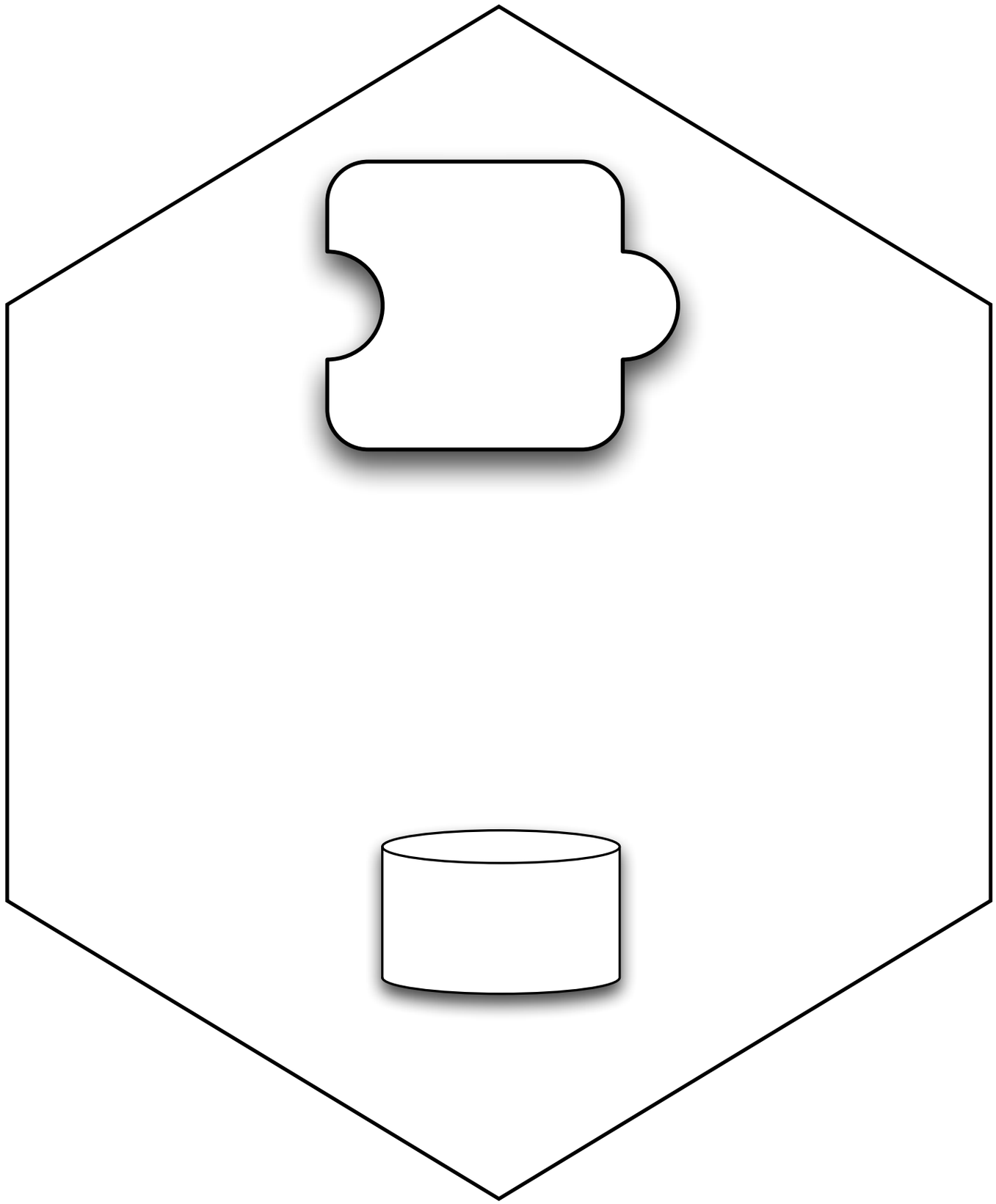

POST /users

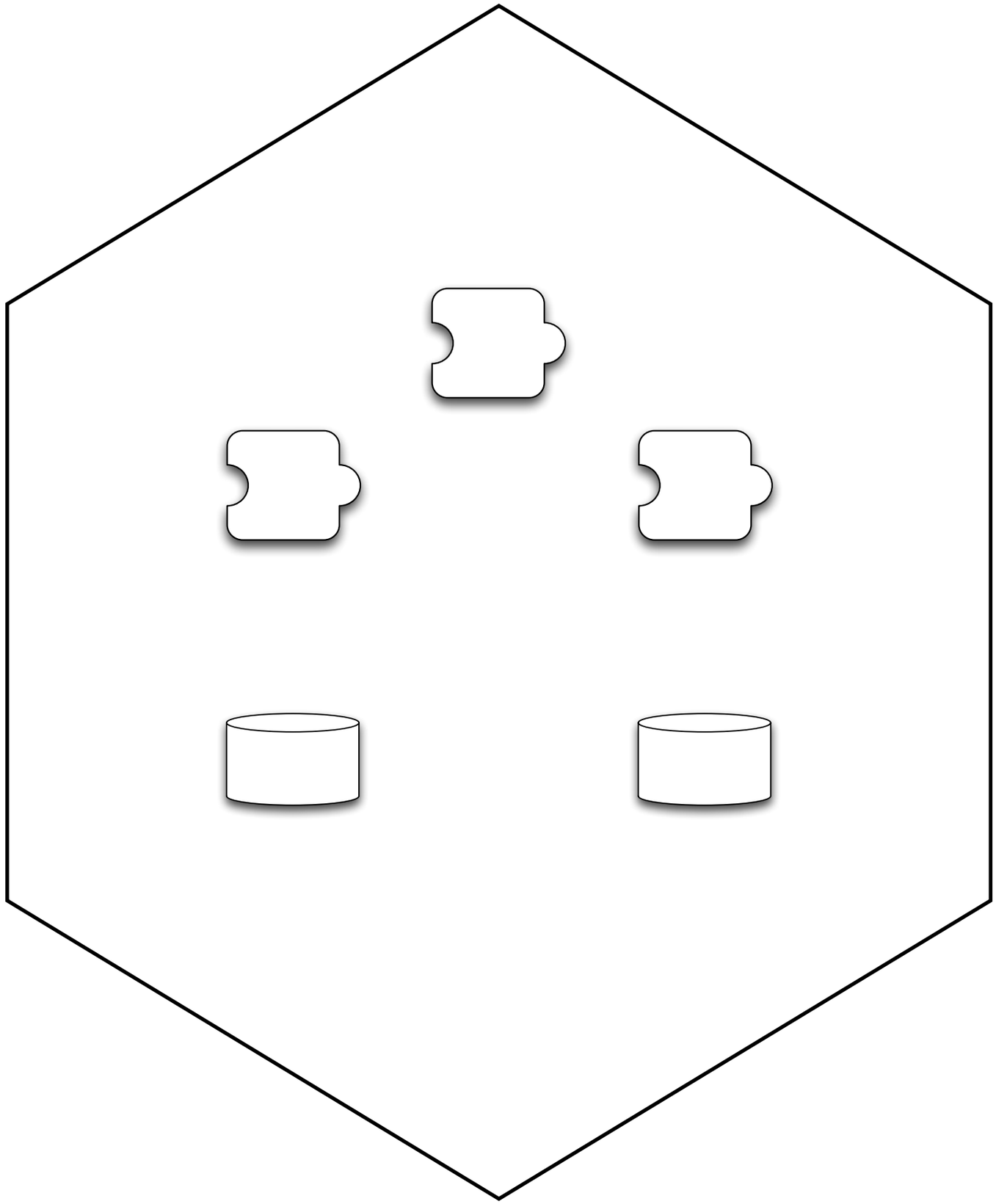


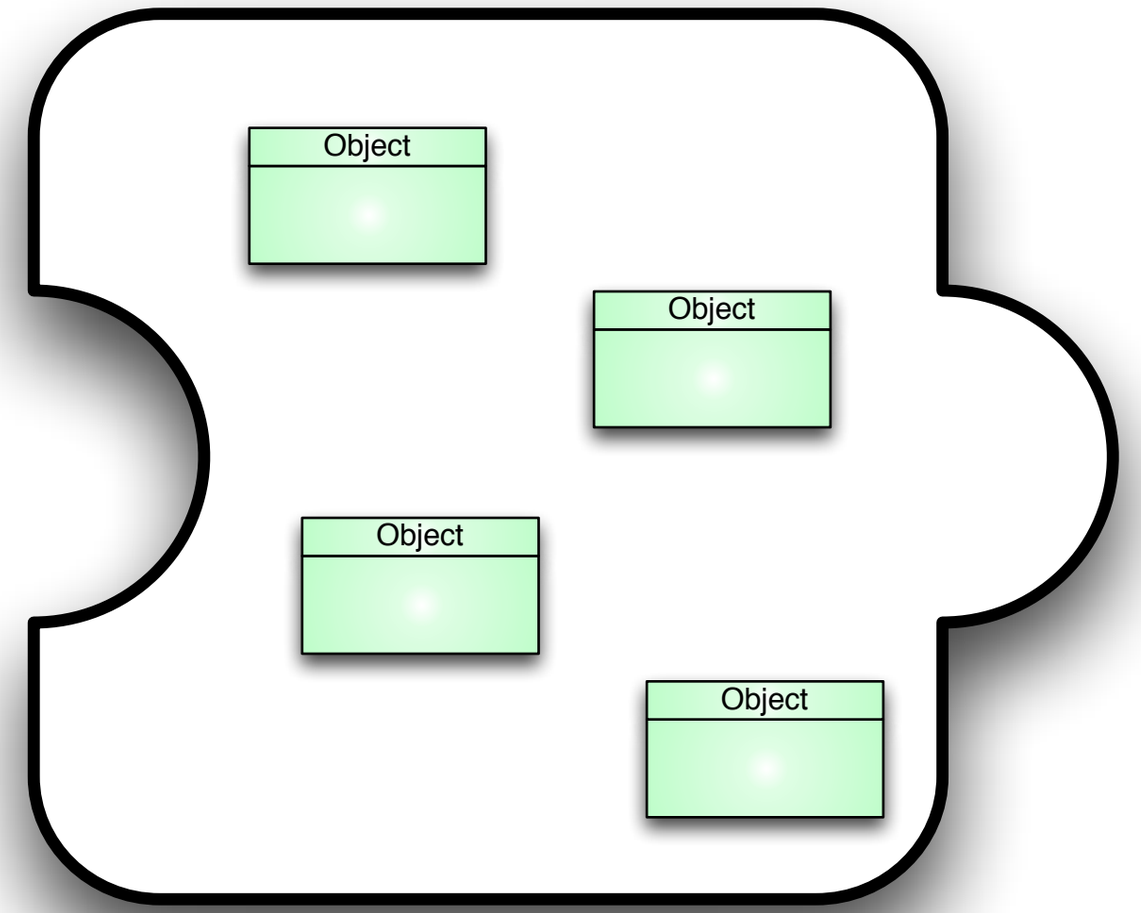

```
POST /users HTTP/1.1
    Host: example.org
    User-Agent: Thingio/1.0
    Authorization: Basic ZGFmZnk6c2VjZXJldA==
    Content-Type: application/atom+xml
    Content-Length: nnn
    Slug: First Post

    <?xml version="1.0" ?>
    <entry xmlns="http://www.w3.org/2005/Atom">
      <title>James Lewis</title>
      <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
      <updated>2003-12-13T18:30:02Z</updated>
      <author><name>James Lewis</name></author>
      <content>stuff</content>
    </entry>
```

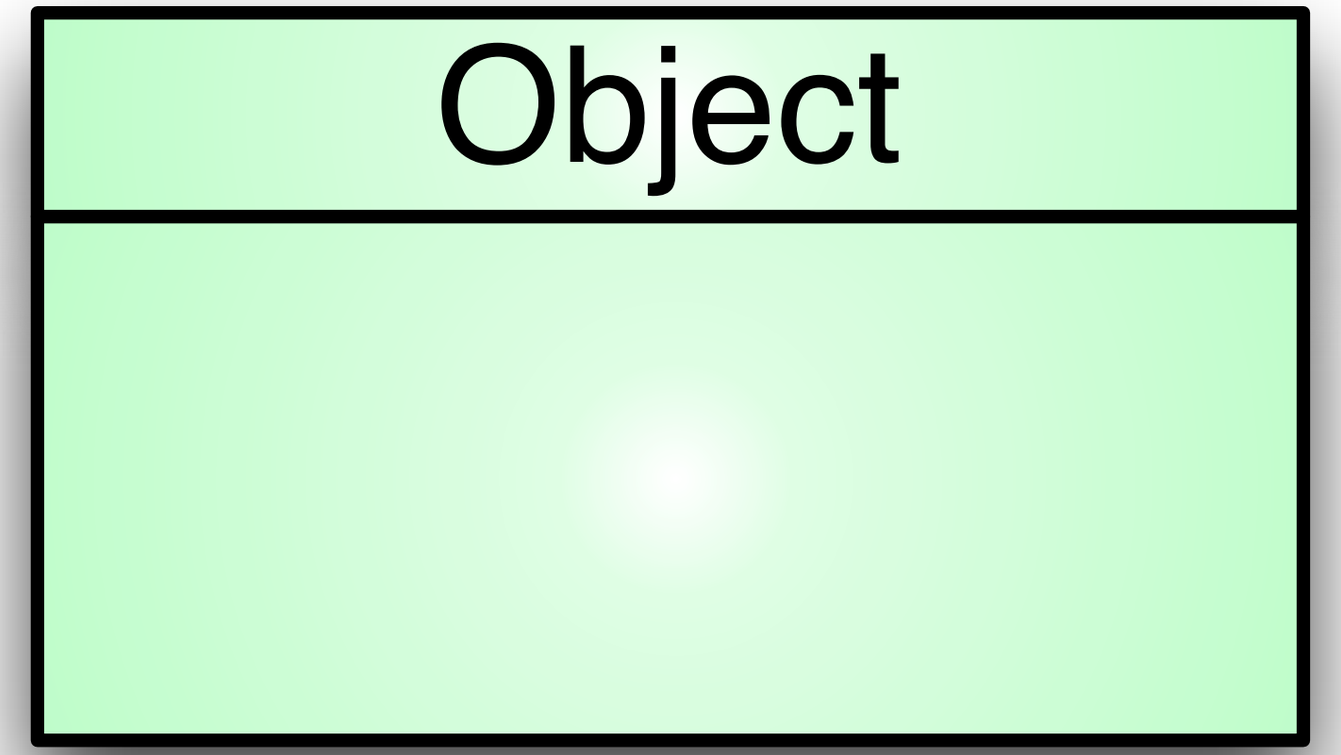

/user-request
/user-request/1223
application/atom+json
EVENT QUEUE
EVENT STORE
PROCESSOR
USER SERVICE
USER STORE

small, with a single responsibility



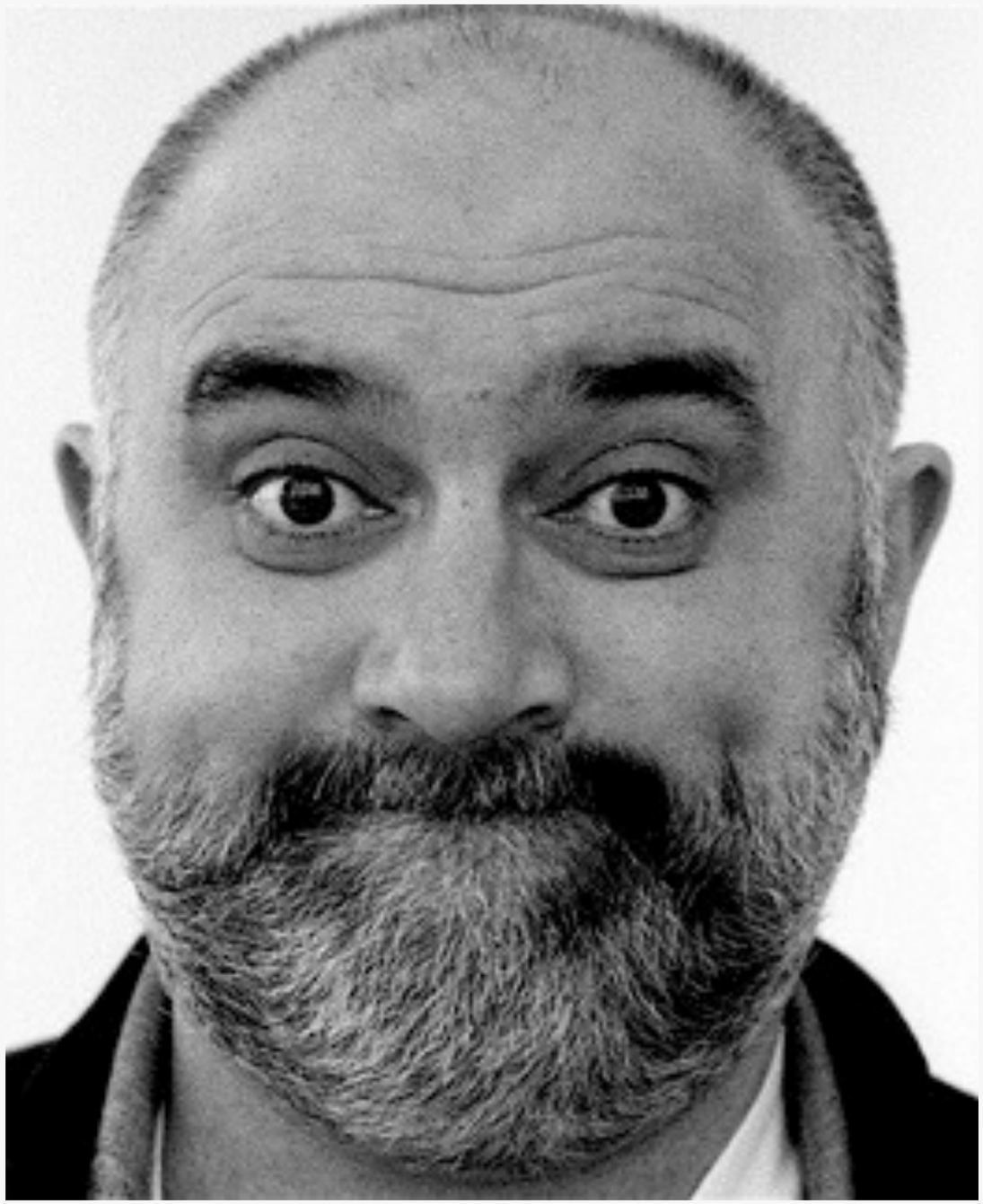



Object
Object
Object
Object

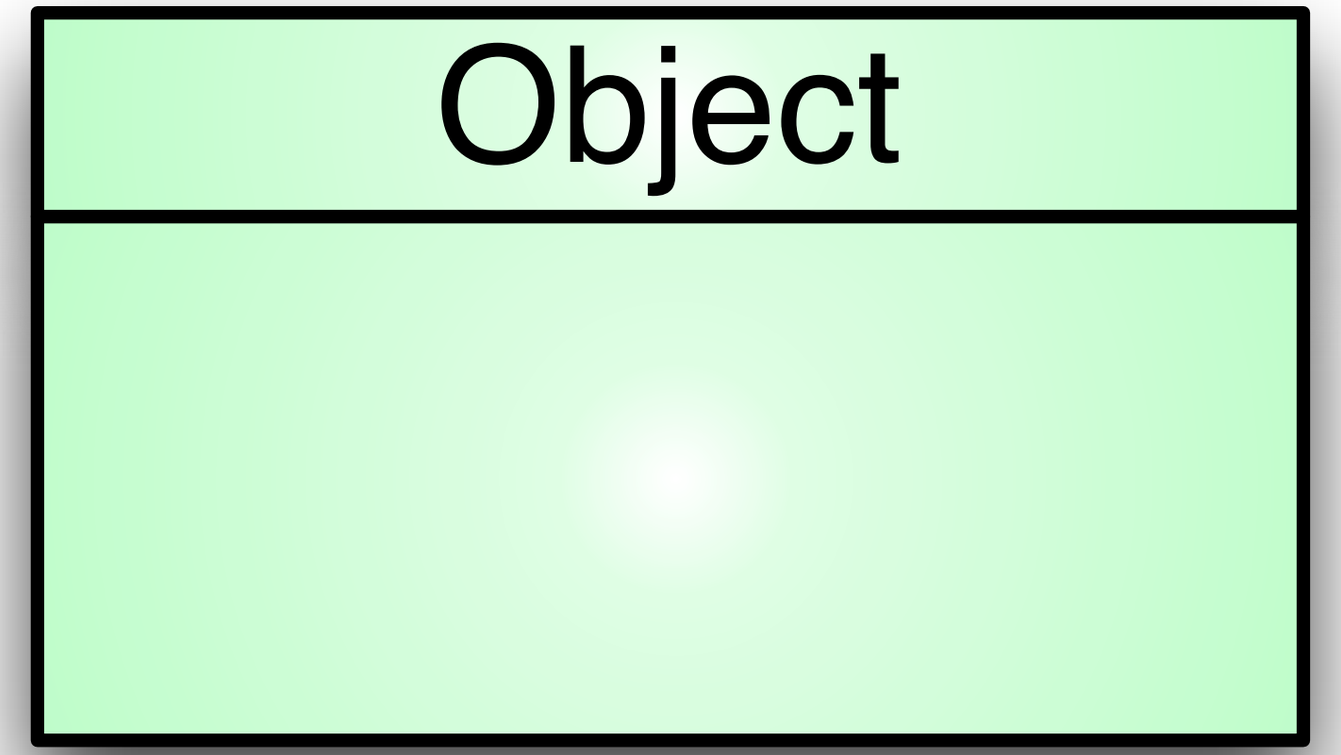


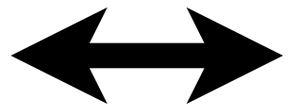
“objects should be no bigger than my head”

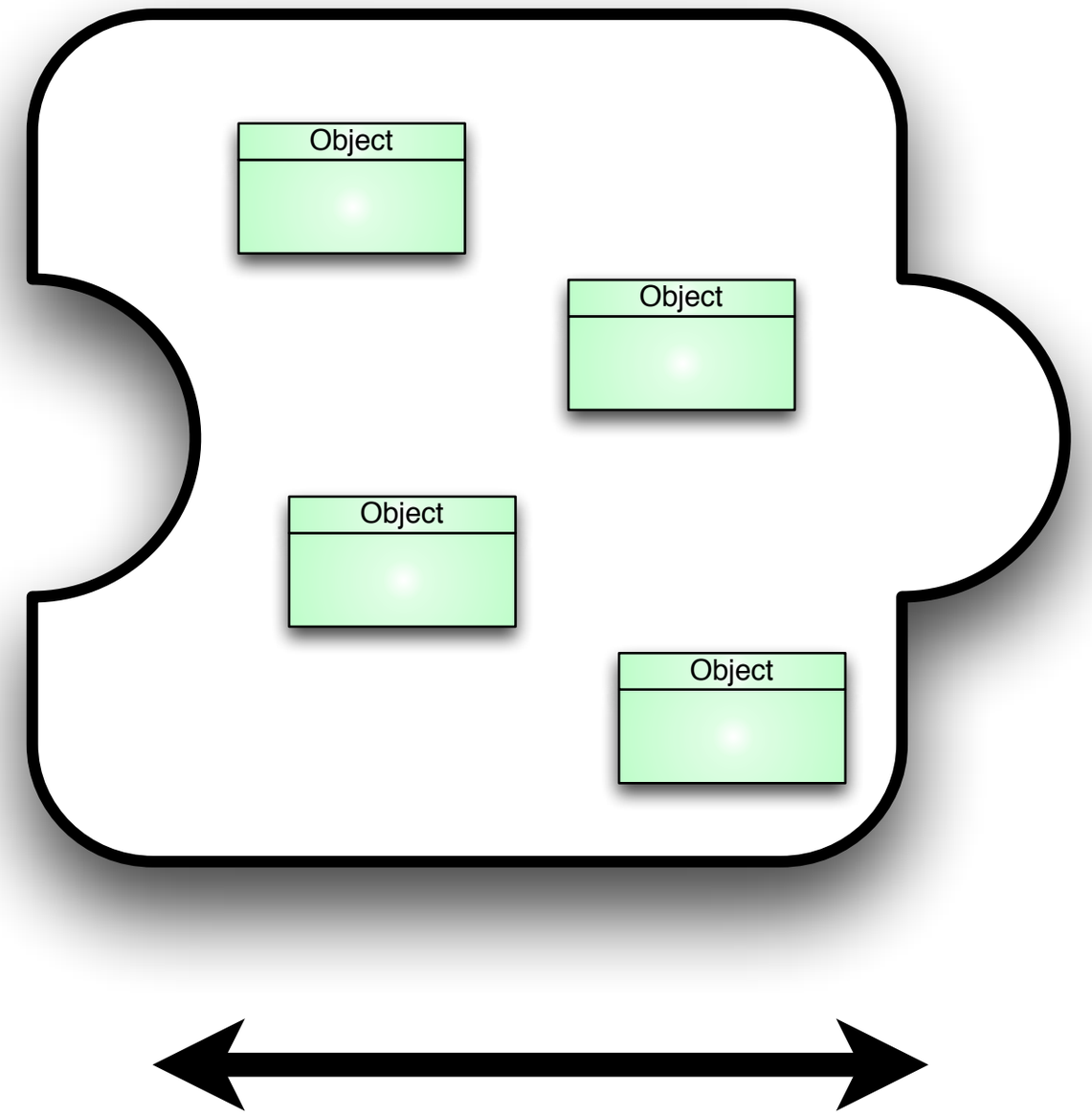
my conjecture

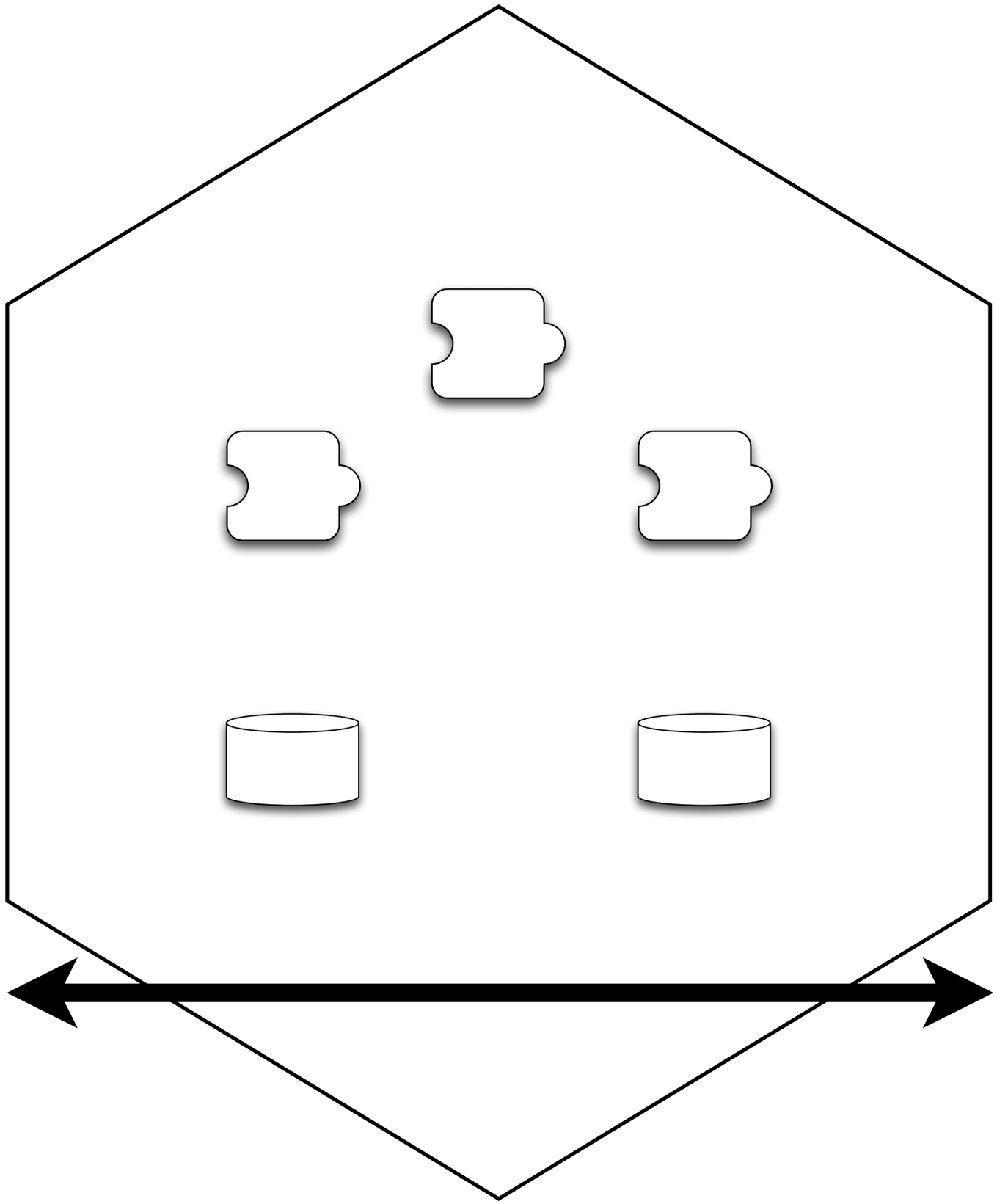


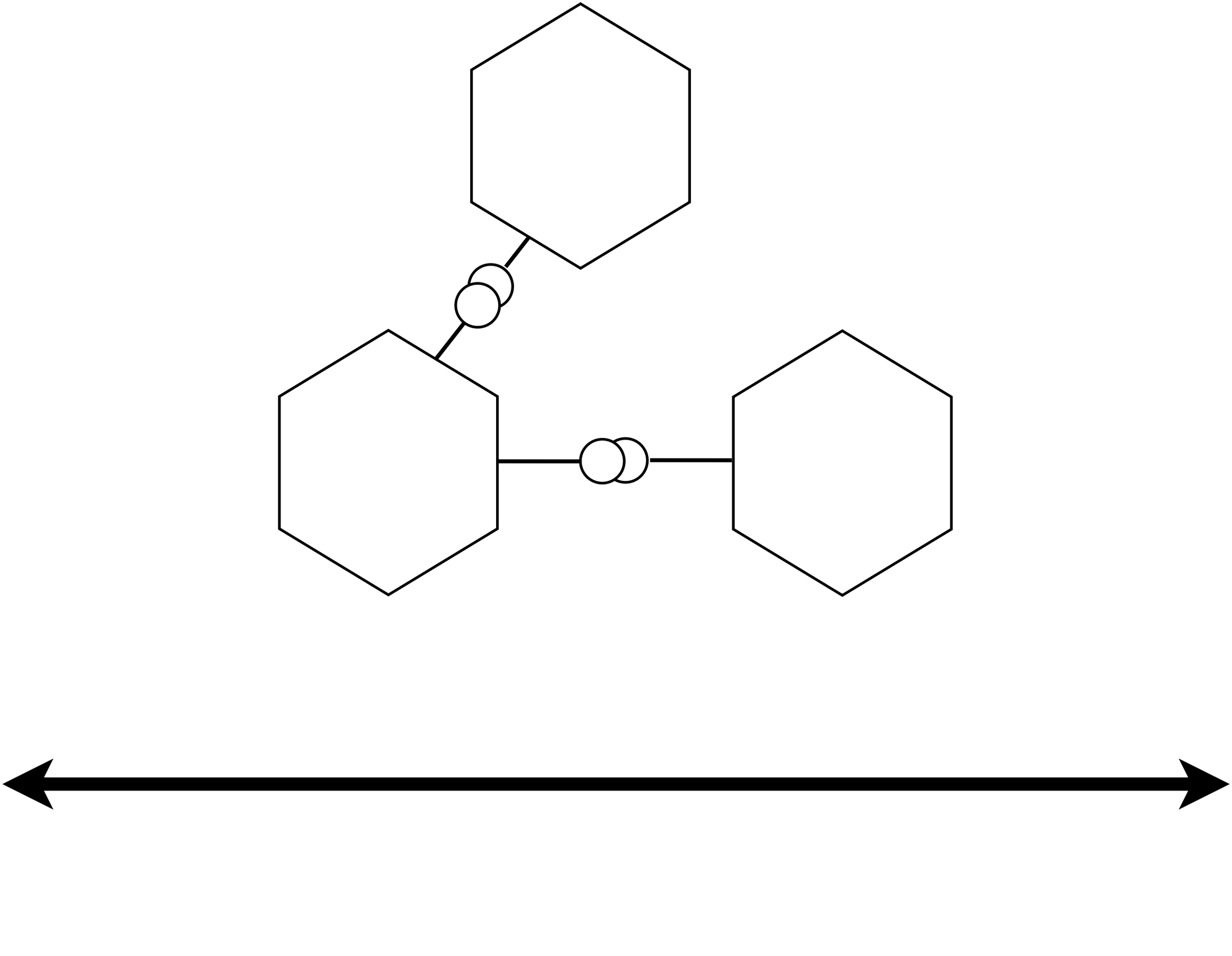
and while I have a giant head, its pretty empty so thats ok...

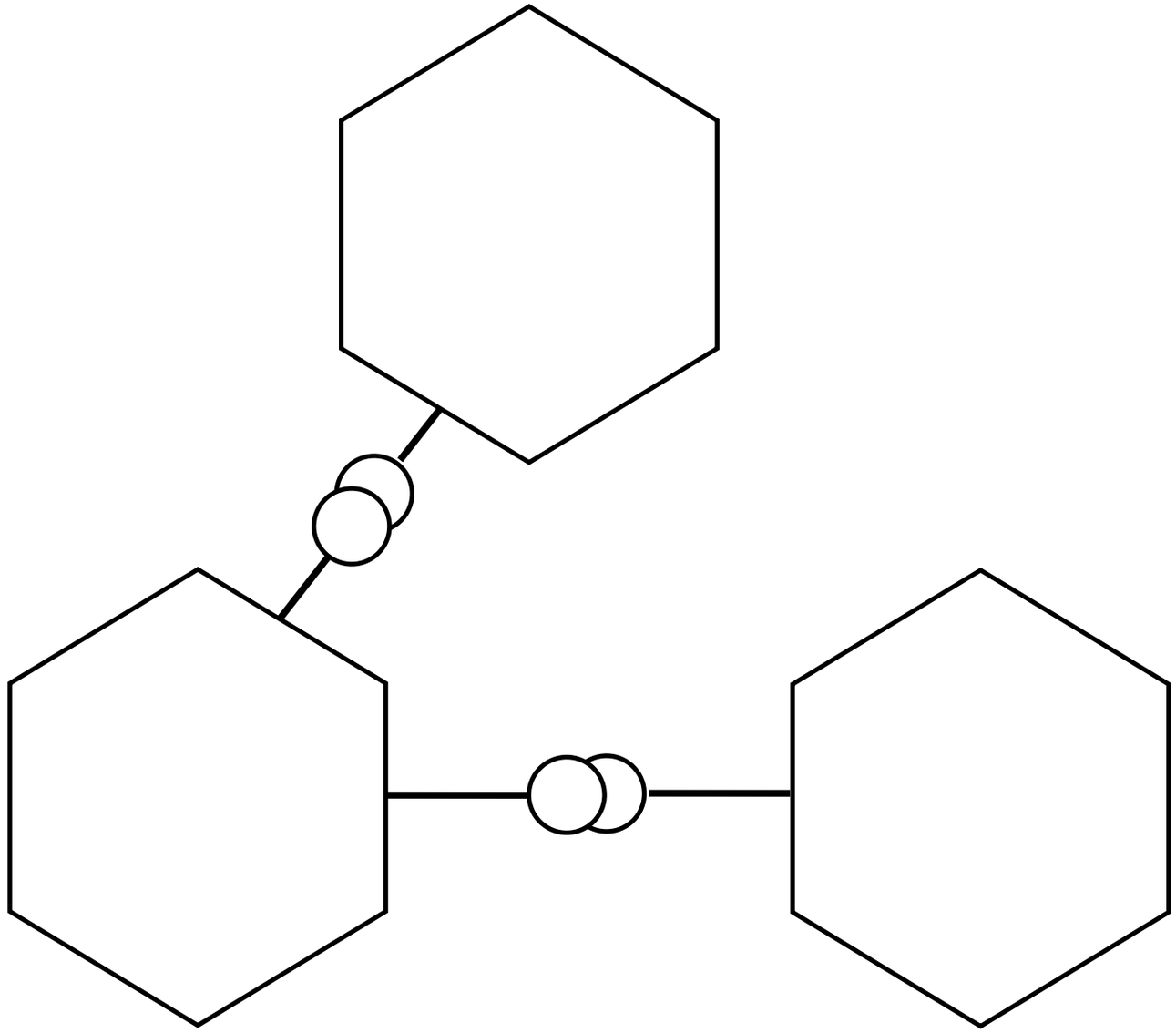

Object




Object
Object
Object
Object

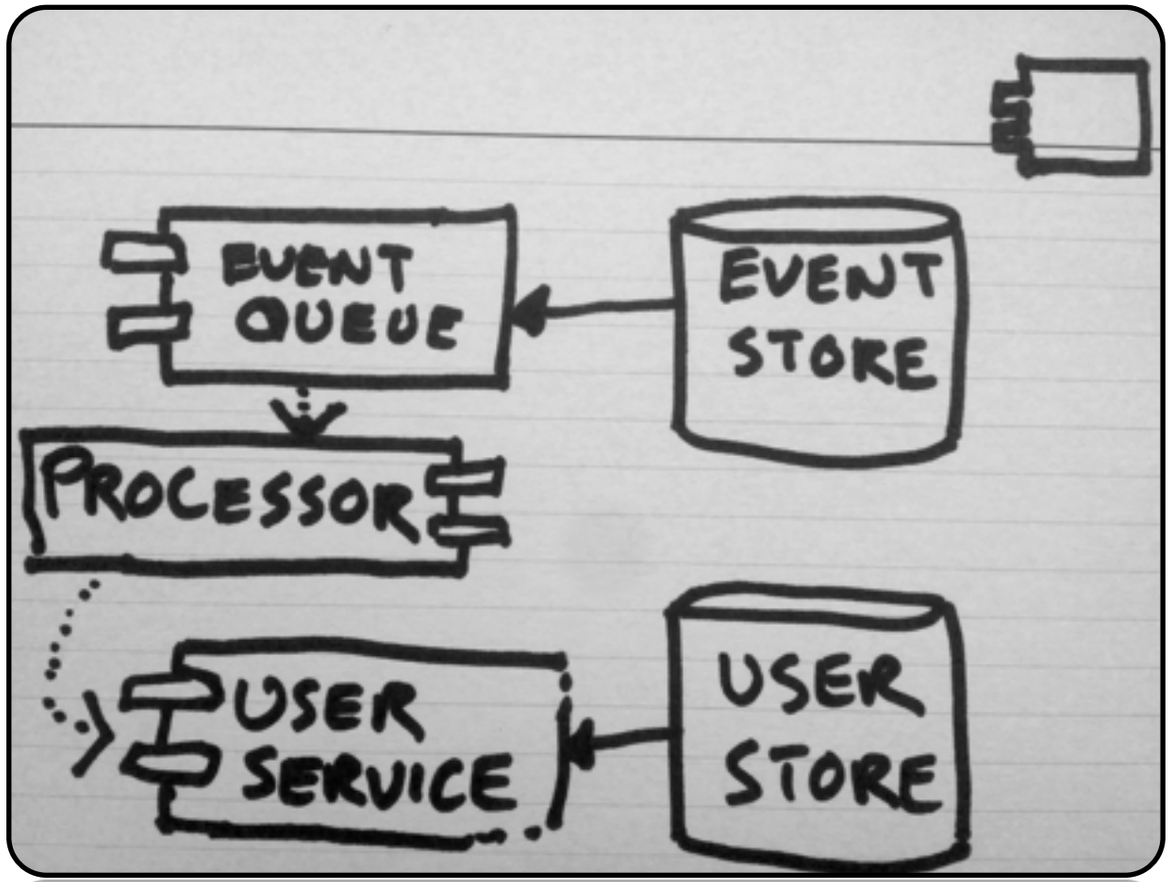






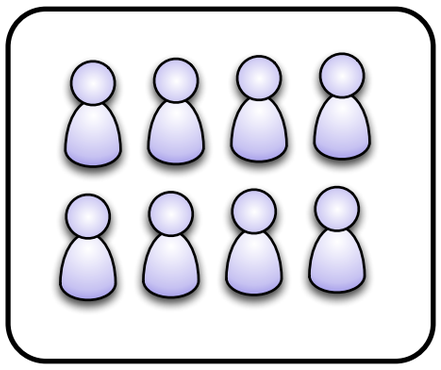
as we chunk up **domains**, each **domain** should be small enough to fit in my head

in this case, it meant 100’s of lines of code per application

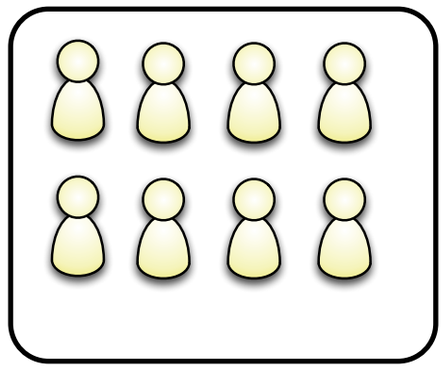


Product Teams organised around Product Lines

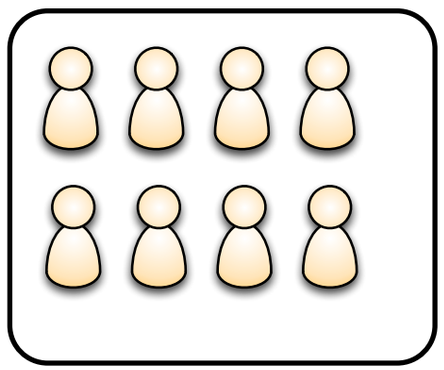
pmo



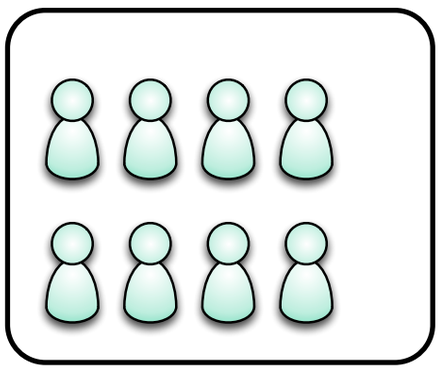
ops



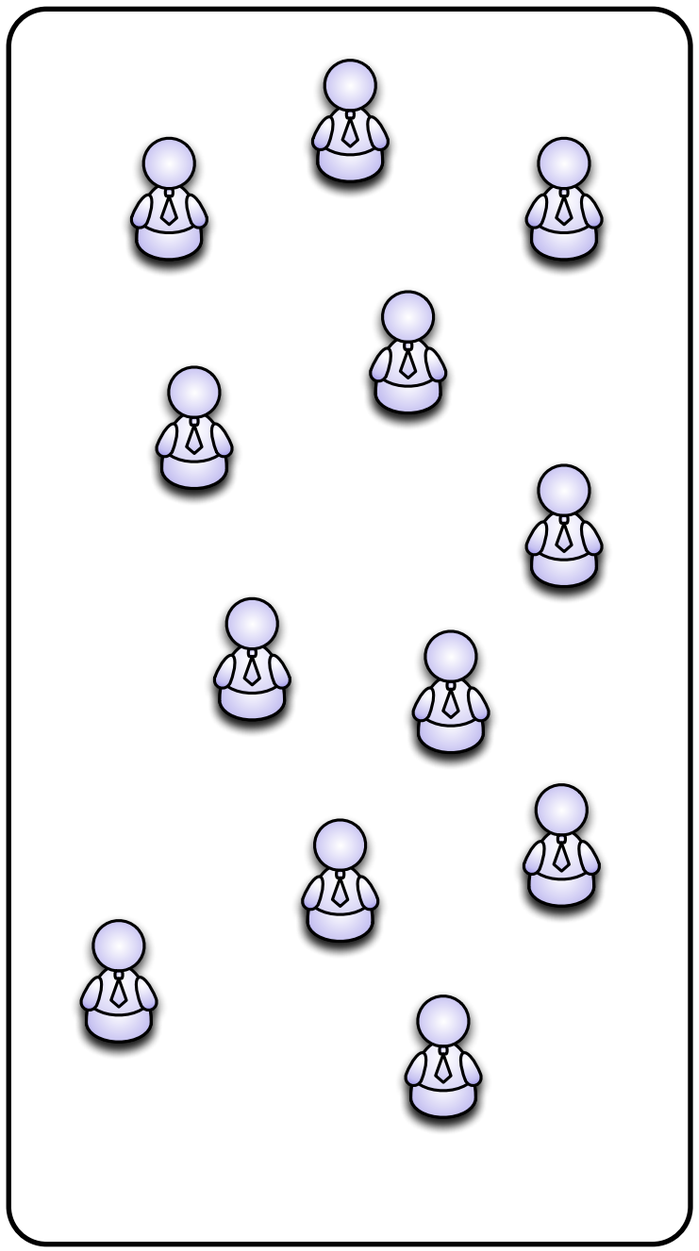
testers



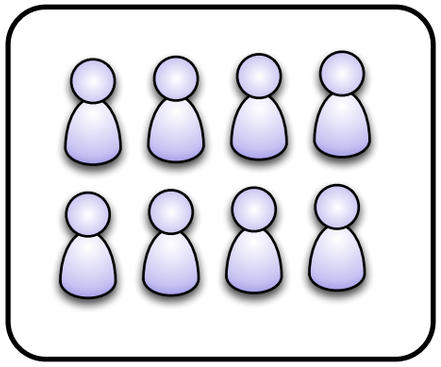
developers



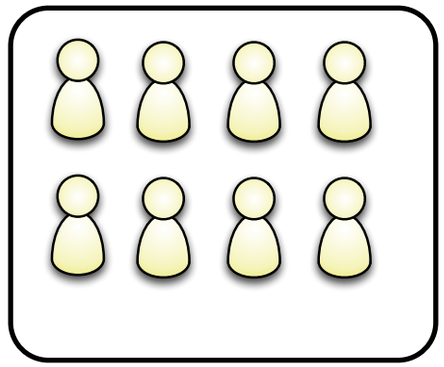
THE BUSINESS



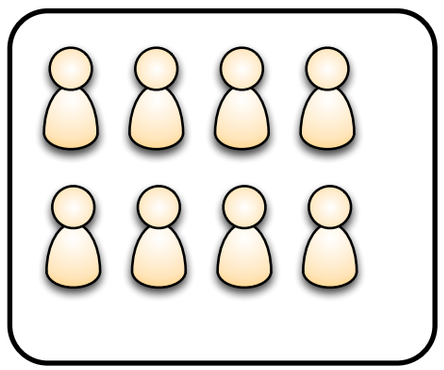
pmo



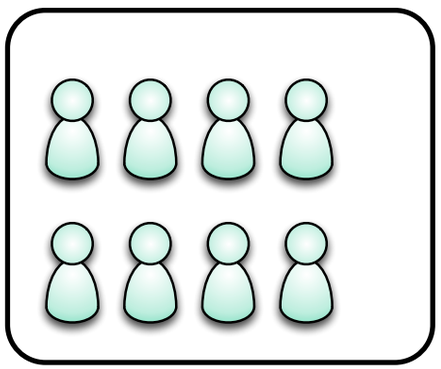
ops



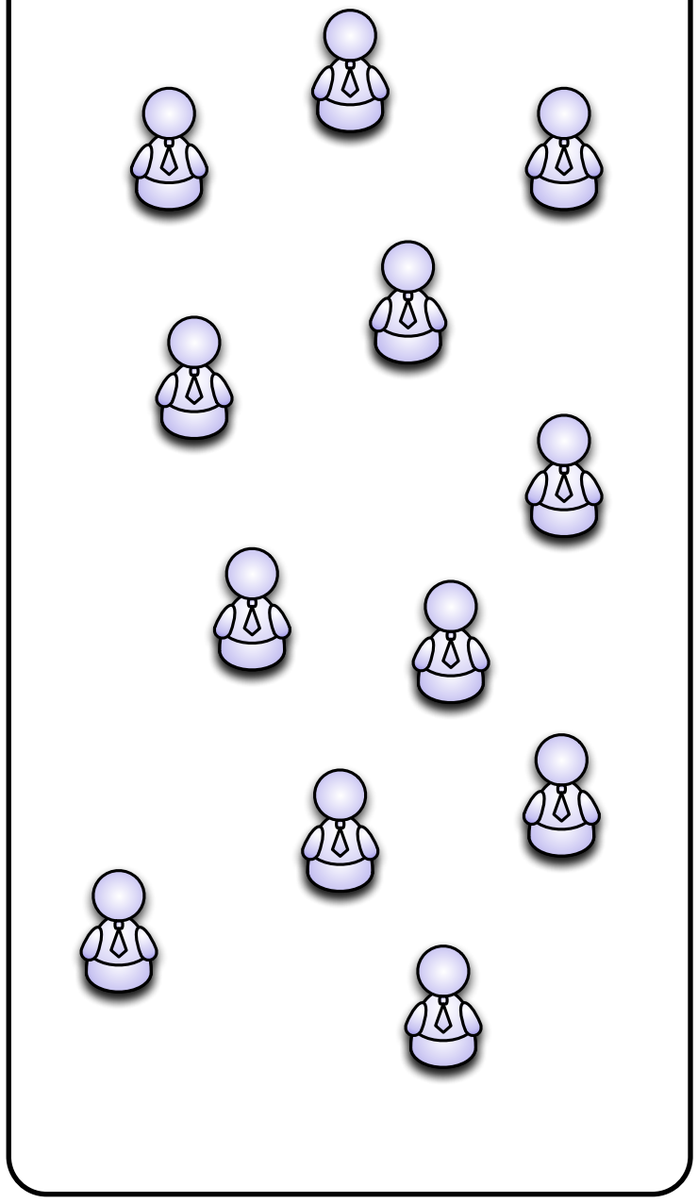
testers

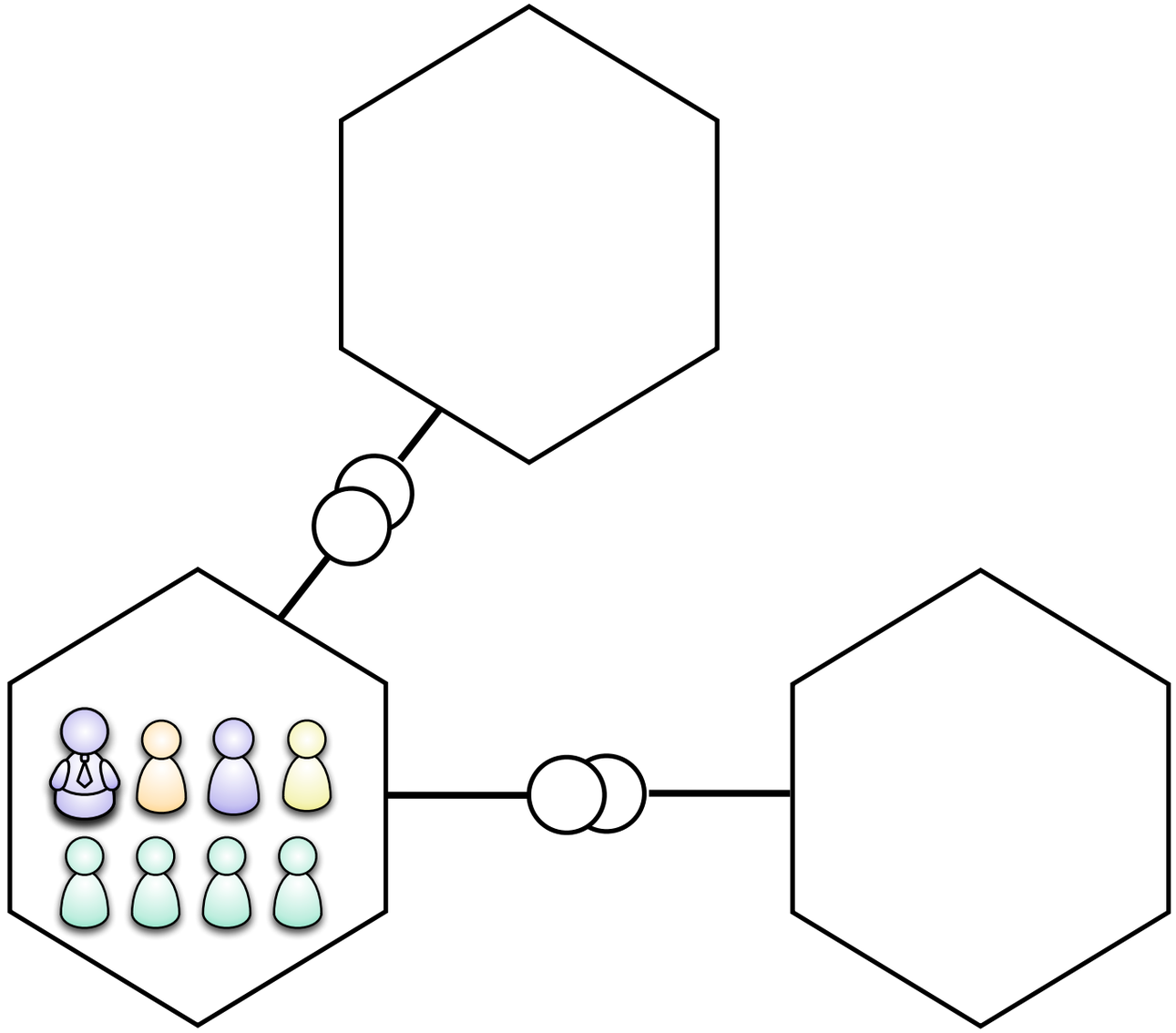


developers



THE BUSINESS





Conway’s Law

Conway’s Law

> “…organizations which design systems … are constrained to produce designs which are copies of the communication structure of those organizations”(sic)

Melvin Conway, 1968

Conway’s Law

> “…organizations which design systems … are constrained to produce designs which are copies of the communication structure of those organizations”(sic)

Melvin Conway, 1968

> “If you ask seven people to write a compiler, you get a seven pass compiler”

Dan North, circa 2006

1 pair

128

2 pairs

243

304

3 pairs

552 + 92 = 644

619

3 pairs

455

346

52

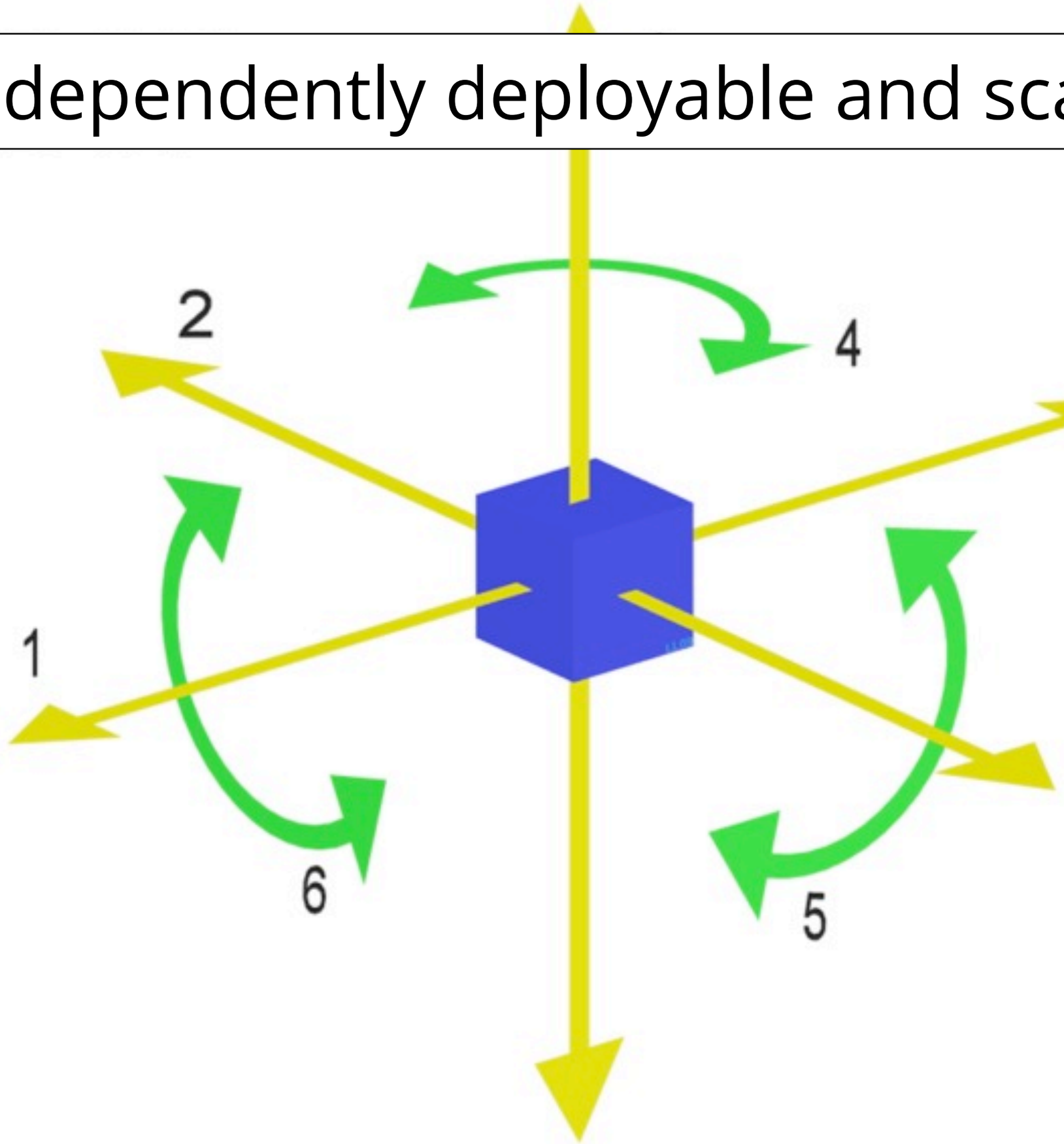
3 pairs

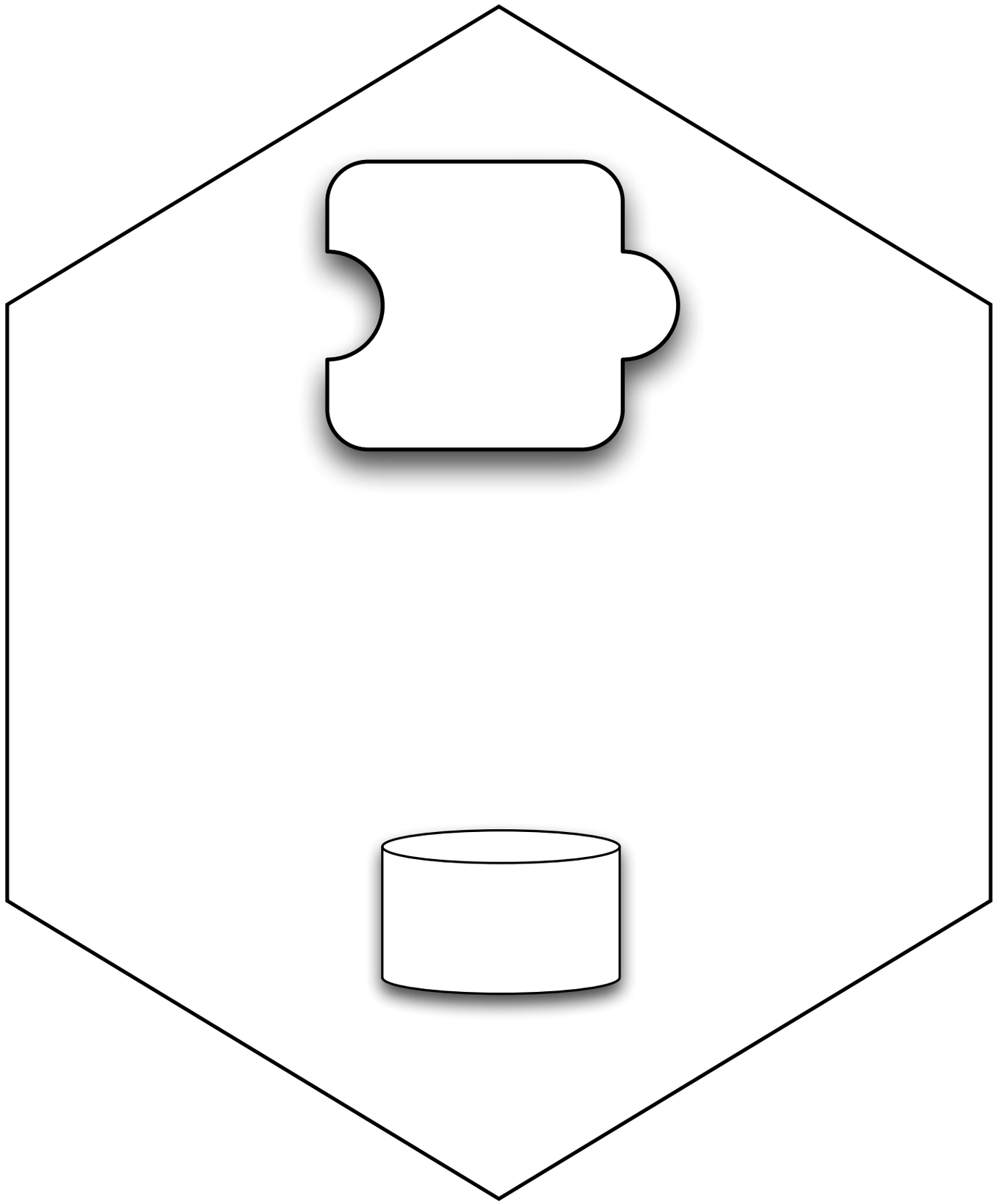
423

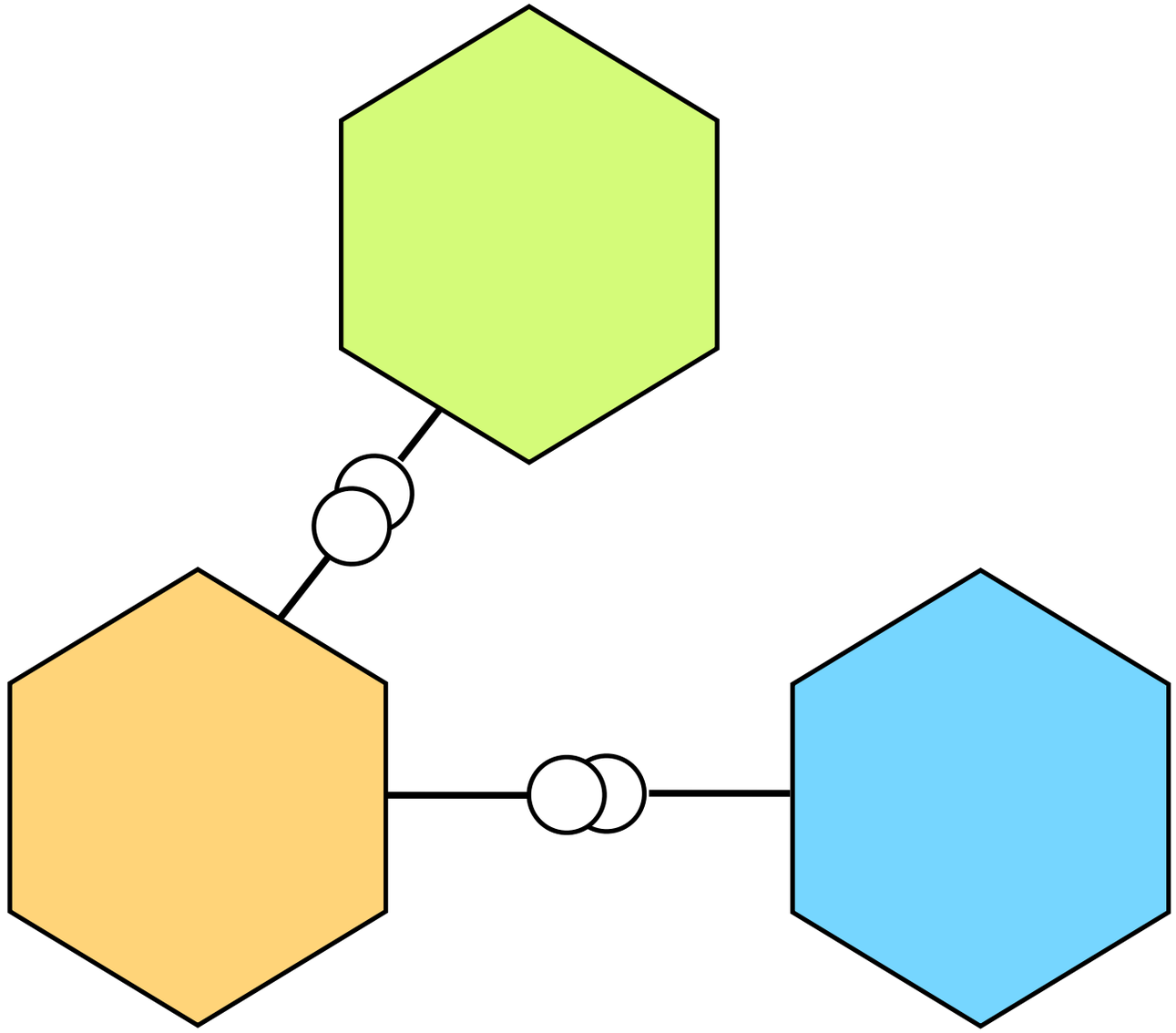
428

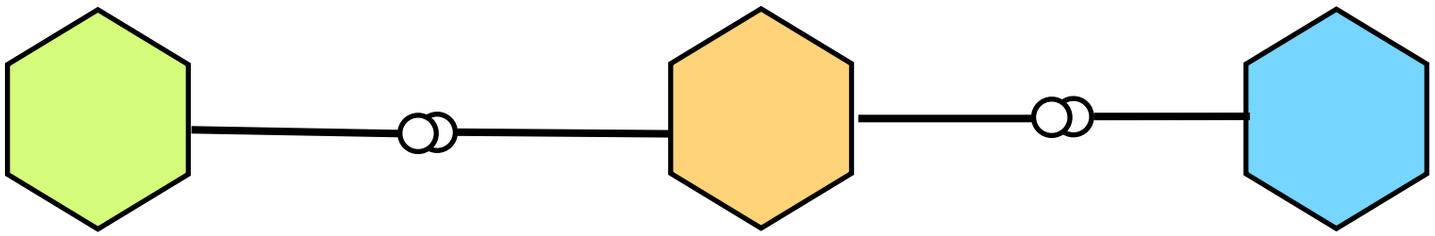
110

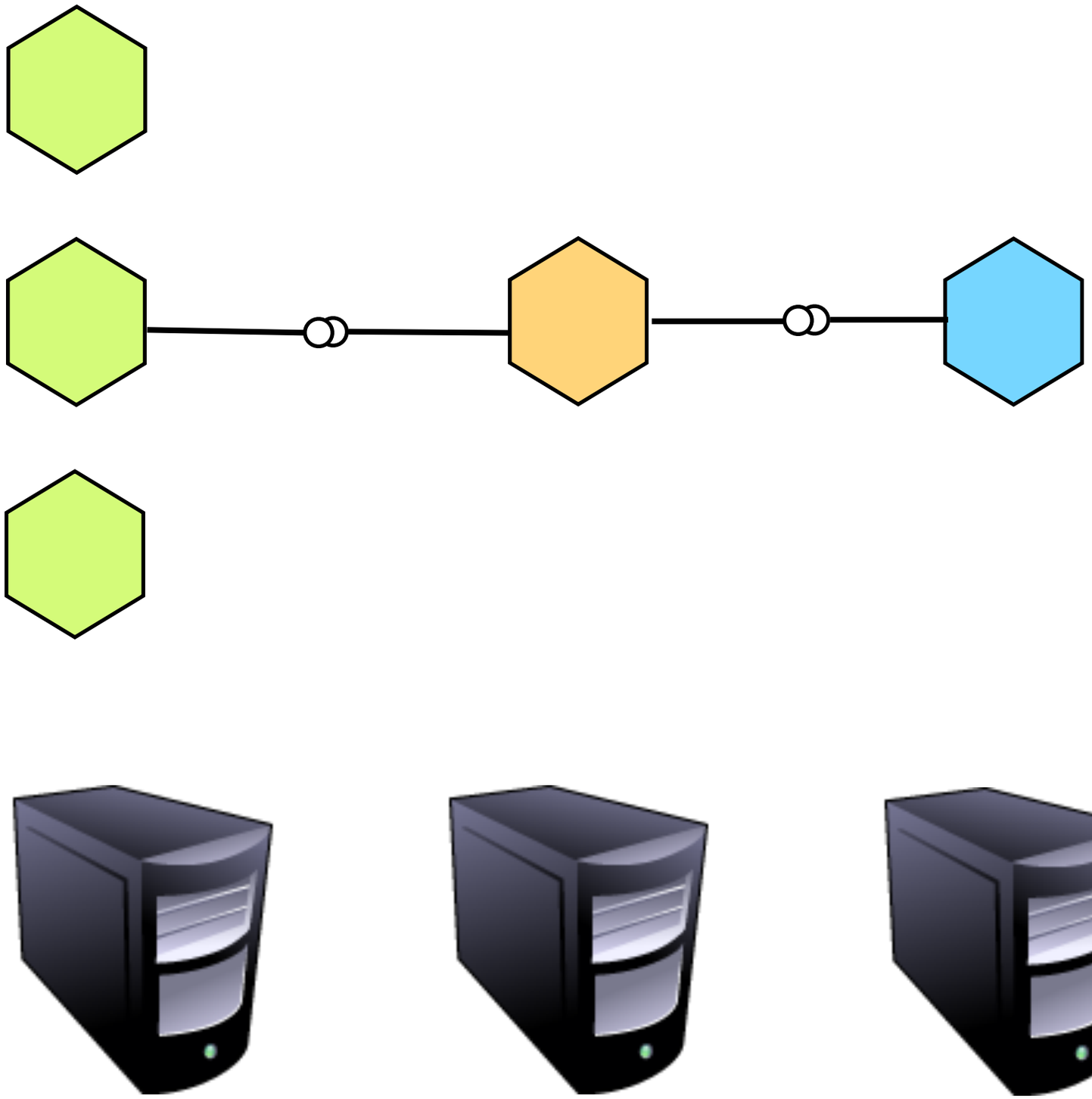
independently deployable and scalable

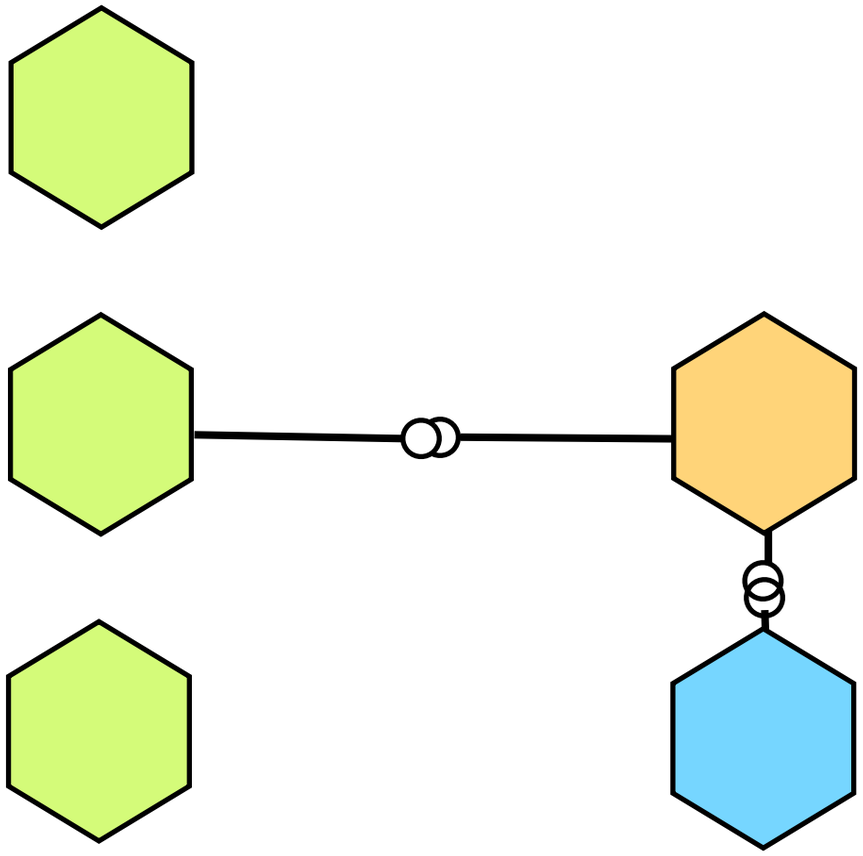






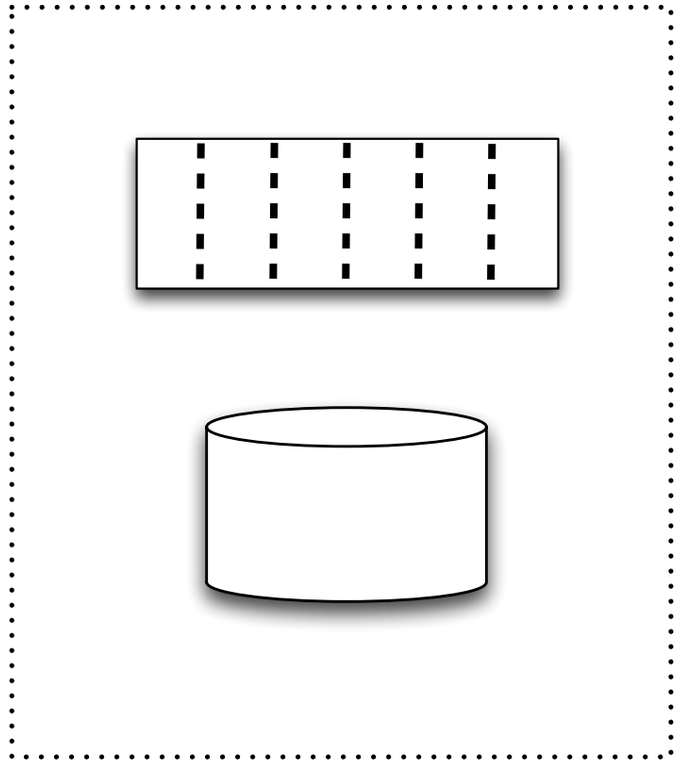




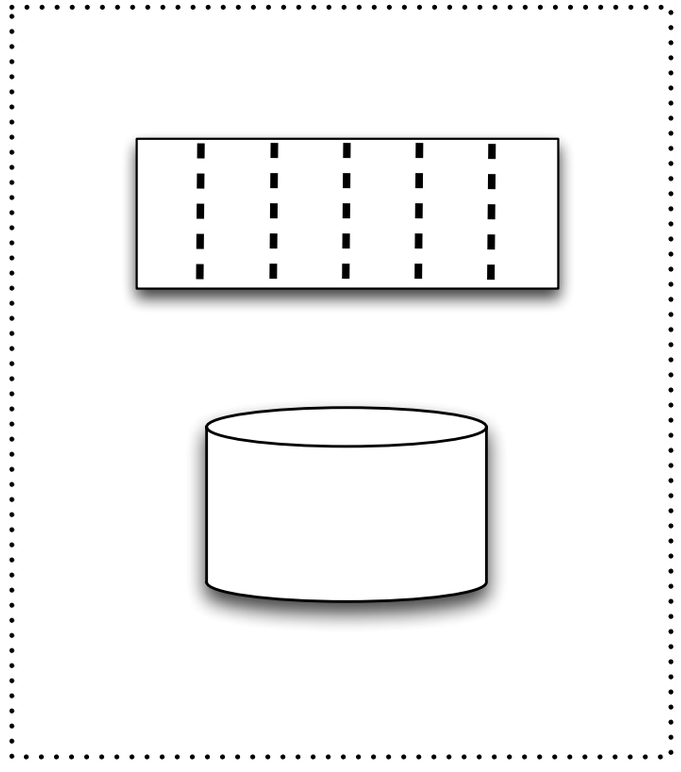


independently scalable, deployable, changeable, replaceable applications

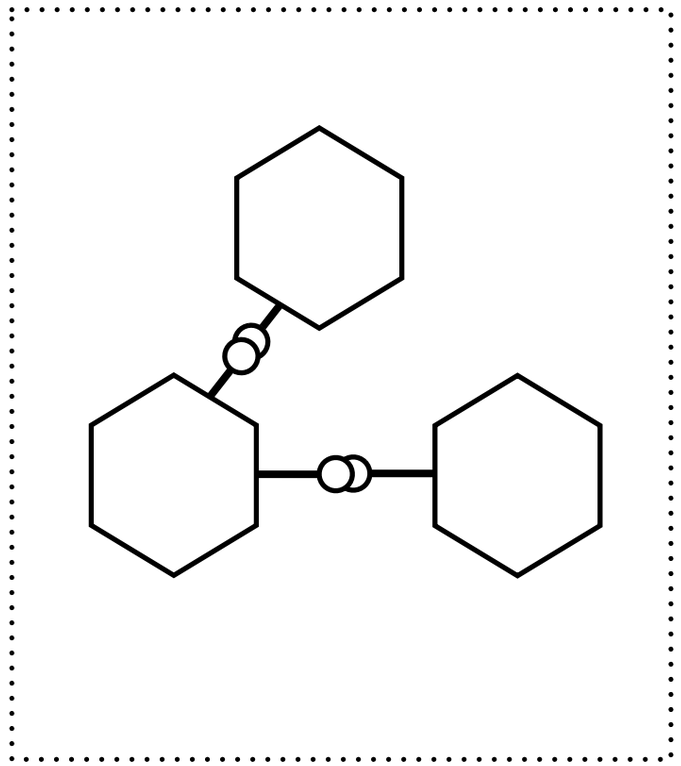
whats the difference between building, deploying and testing this:



whats the difference between building, deploying and testing this:



and this?



~68% said yes

~ 32% said no

~80% are using automated provisioning

~20%

independently scalable, deployable, changeable, replaceable applications

independently scalable, deployable, changeable, replaceable applications

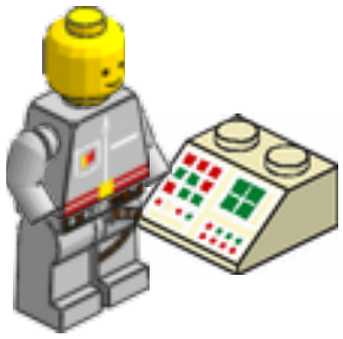
built, deployed and scaled automatically

building micro-services

a basic build pipeline

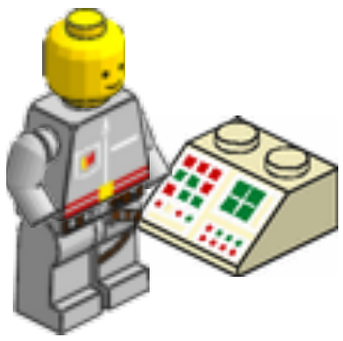
a basic build pipeline

hey, it works on my machine!



a basic build pipeline

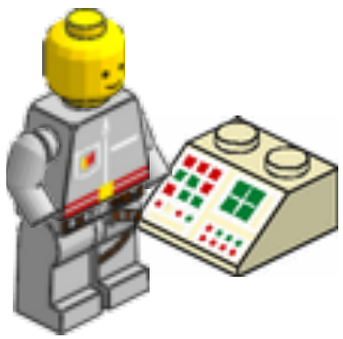
hey, it works on my machine!



I GUESS ITS TIME TO COMMIT AND PUSH

a basic build pipeline

hey, it works on my machine!



I GUESS ITS TIME TO COMMIT AND PUSH

compile and unit test

integration test

performance test

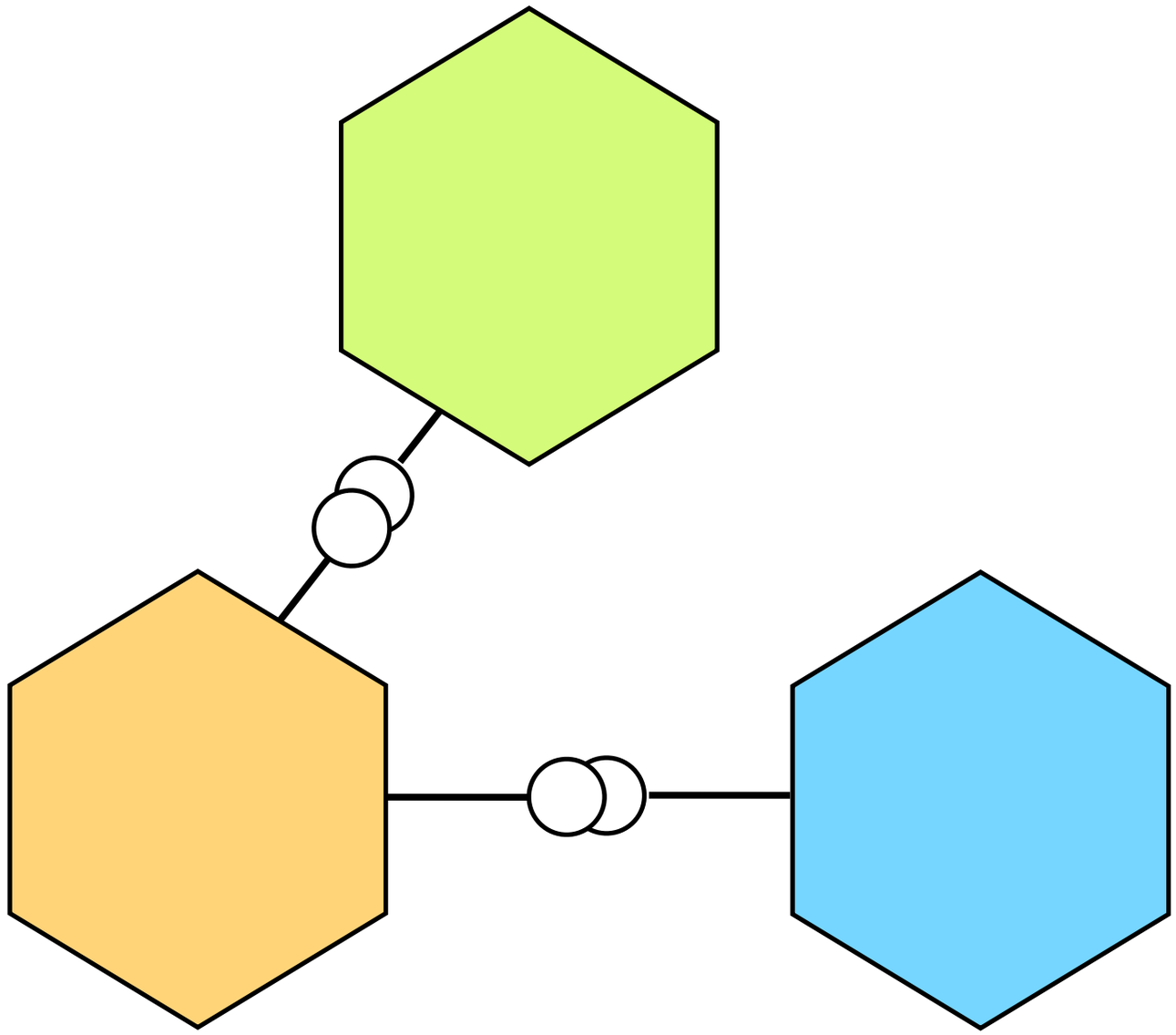
acceptance test

UAT

deploy to production

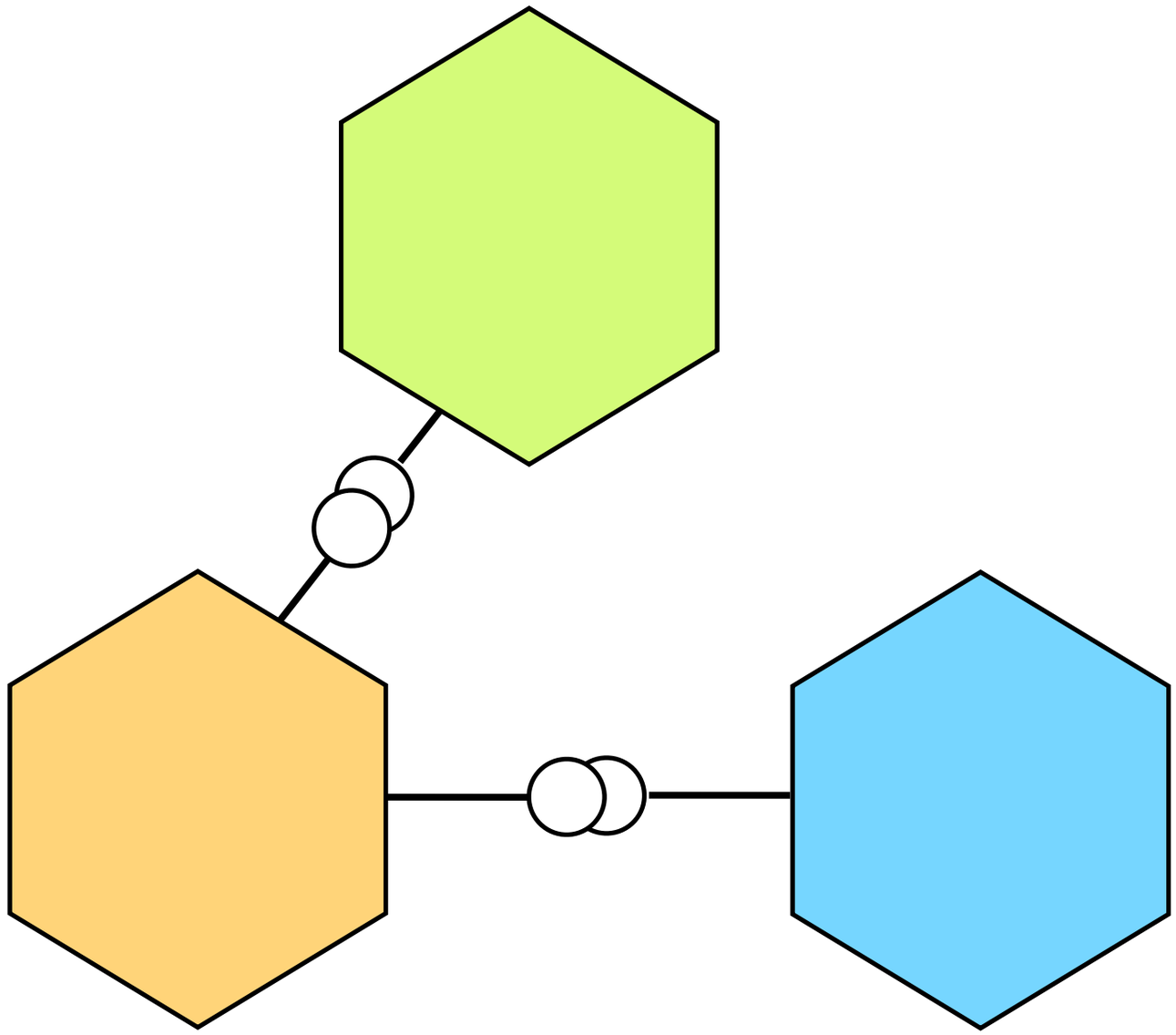
if you have lots of small applications all doing one thing

observing other resources, say



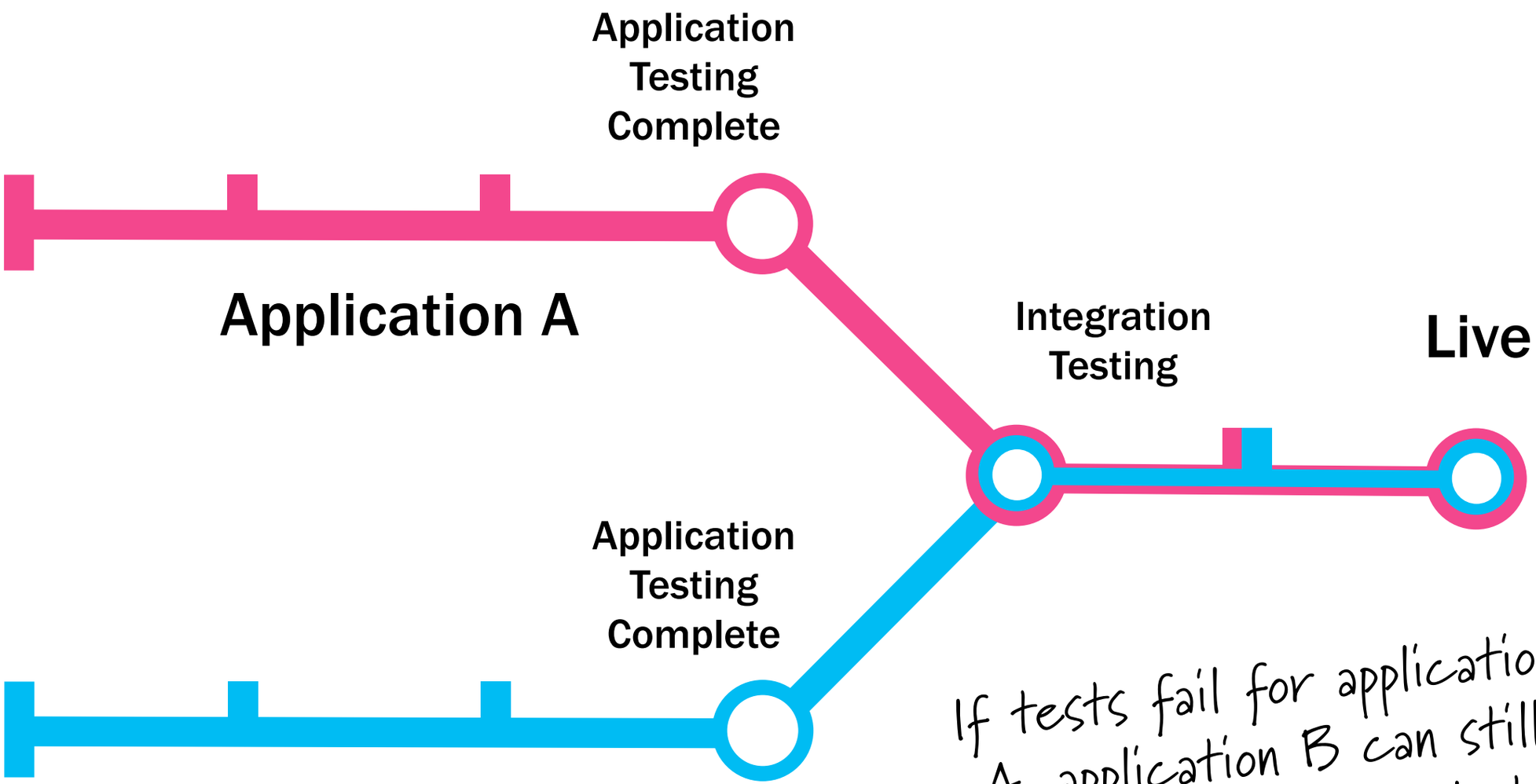
if you have lots of small applications all doing one thing

observing other resources, say



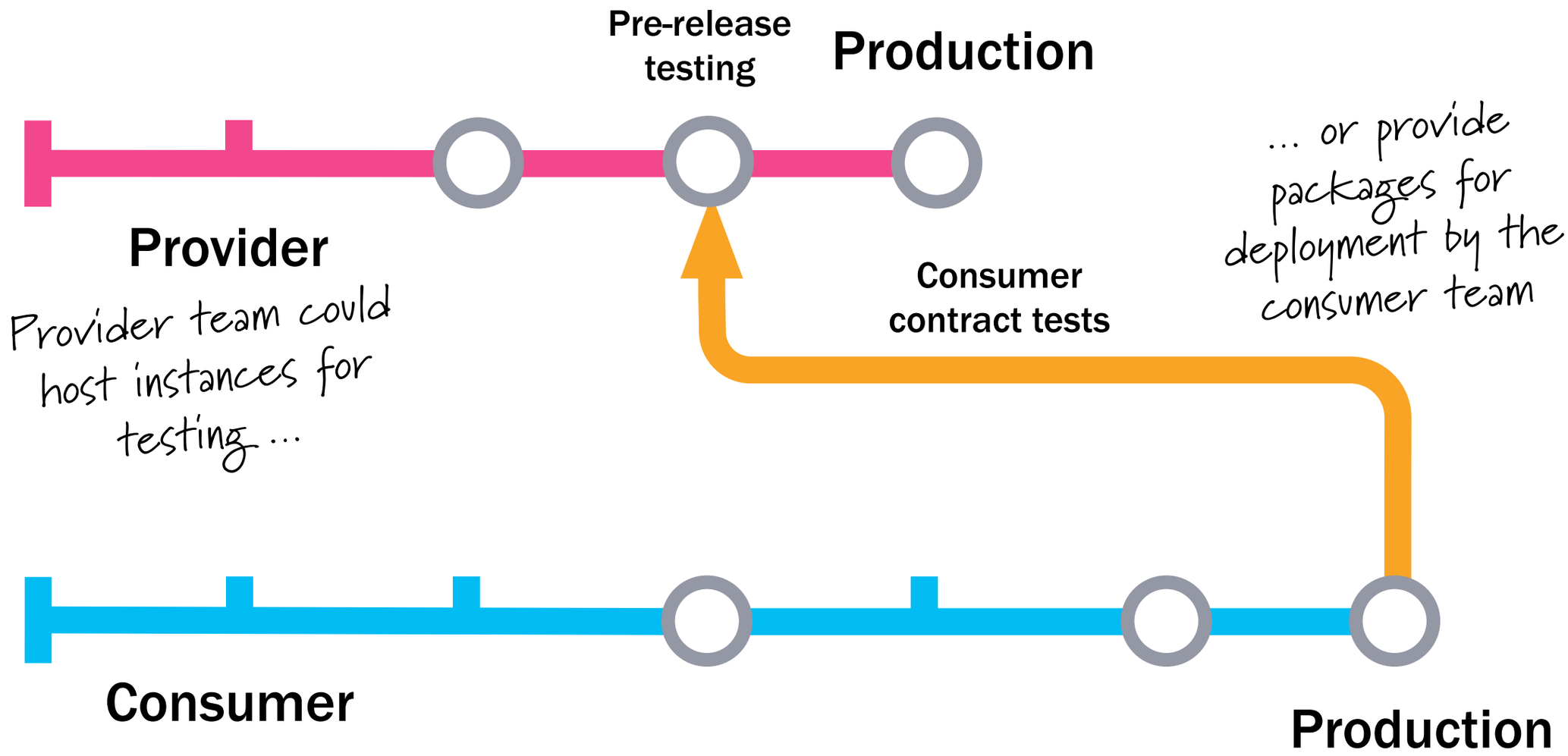
it is harder to put together a build pipeline for the components

Joining multiple applications

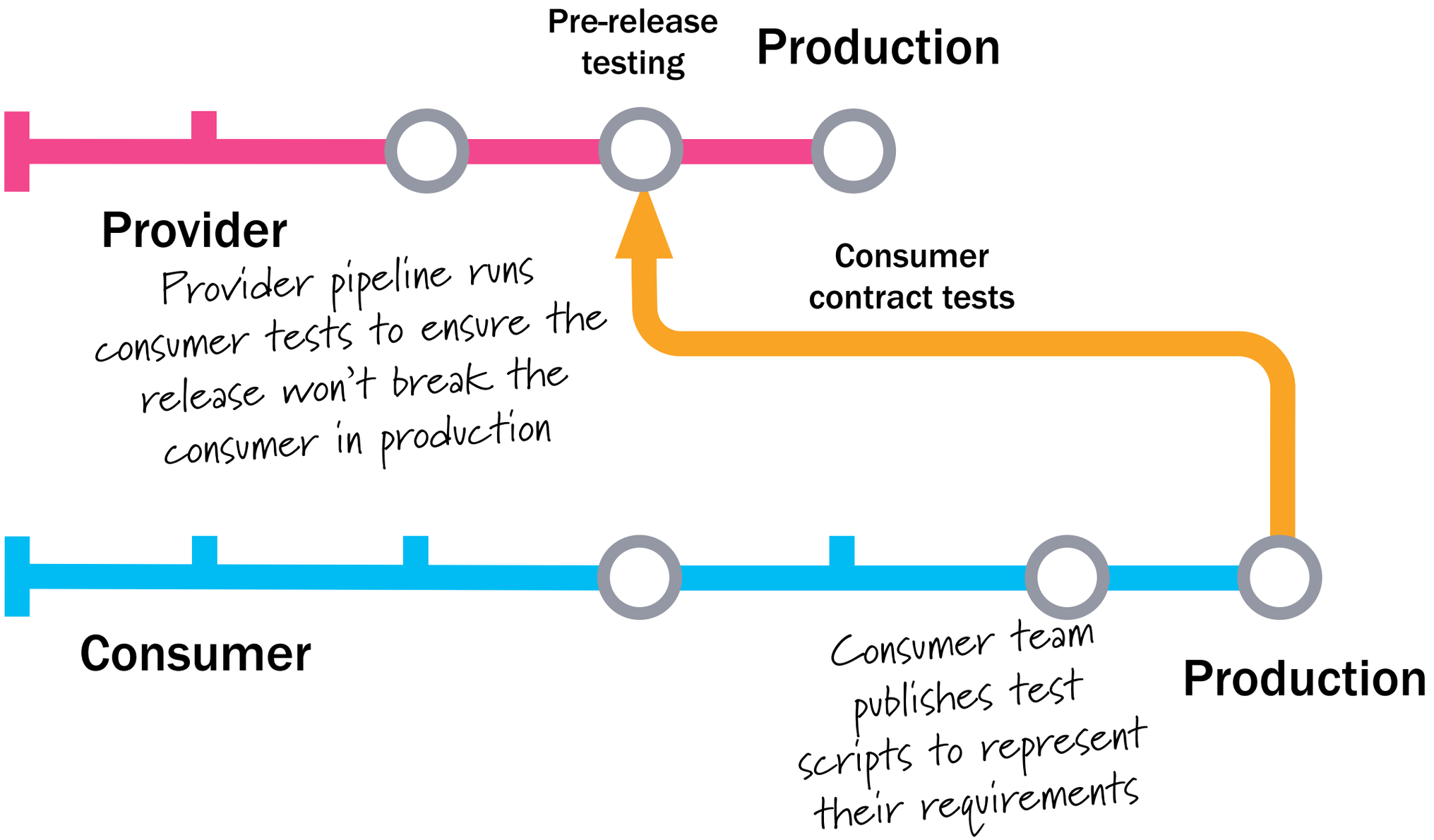


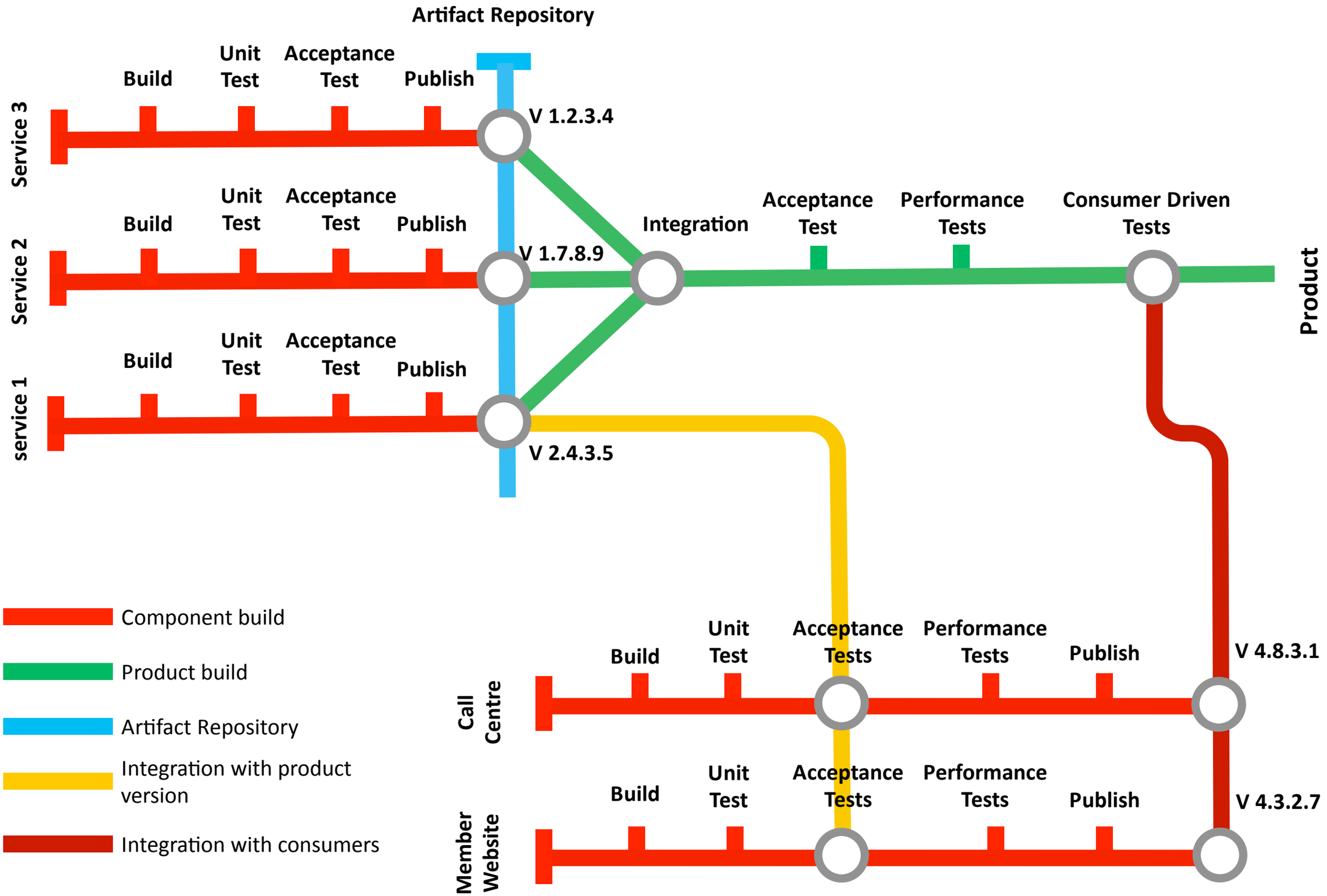
If tests fail for application A, application B can still proceed using the last "good" version of A

Federated pipelines

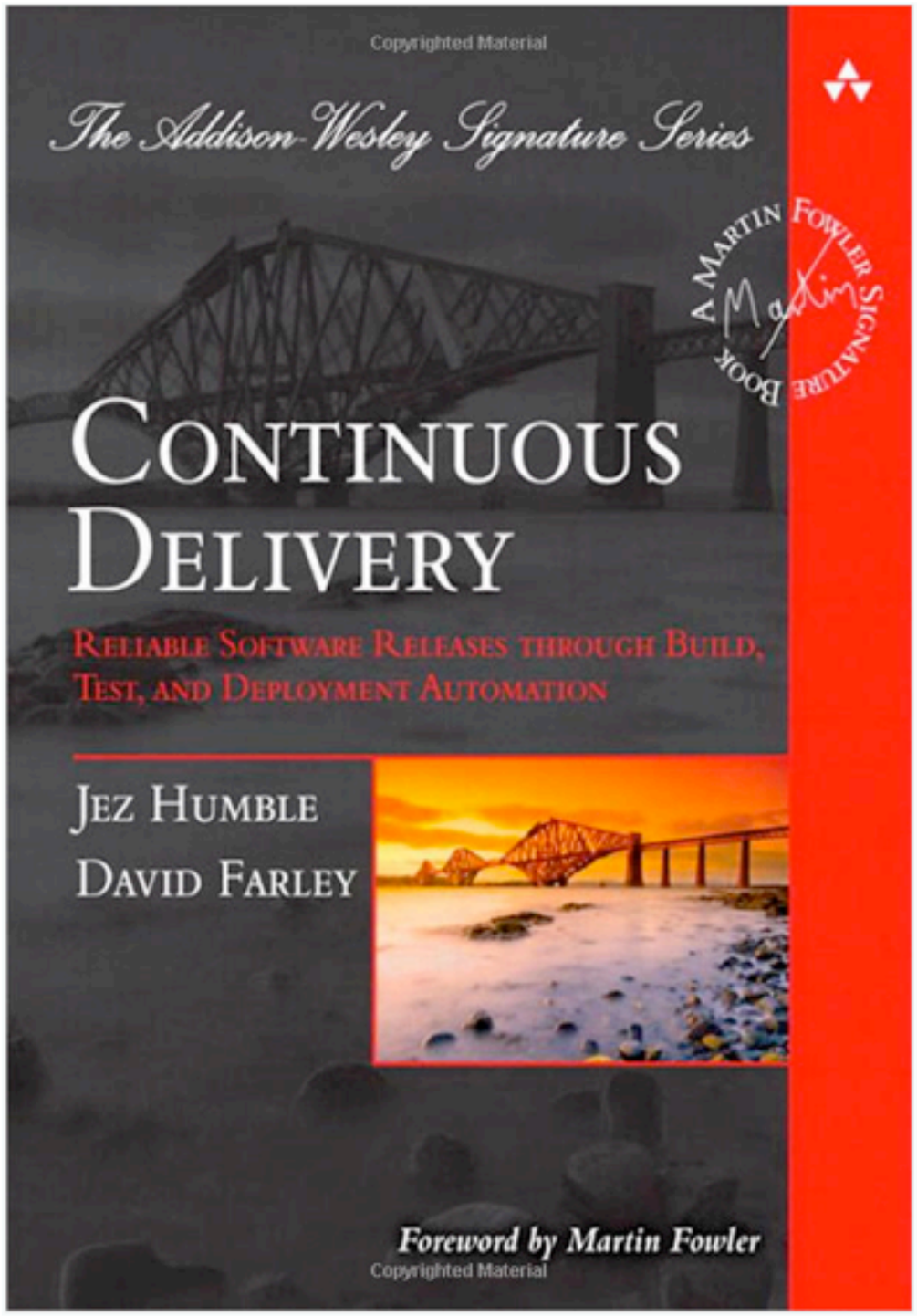


Federated pipelines




Artifact Repository
Build
Unit Test
Acceptance Test
Publish
Service 3
V 1.2.3.4
Service 2
V 1.7.8.9
service 1
V 2.4.3.5
Integration
Acceptance Test
Performance Tests
Consumer Driven Tests
Product
Component build
Product build
Artifact Repository
Integration with product version
Integration with consumers
Call Centre
Acceptance Tests
V 4.8.3.1
Member Website
V 4.3.2.7

there is a little bit more in here...



deploying many services

infrastructure automation

phoenix infrastructure

autoscaling and status aware

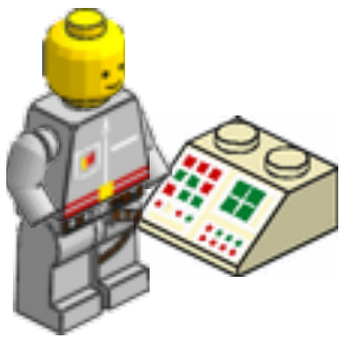
ZOMBIE APOCALYPSE

PRESS FIRE TO PLAY

we saw earlier the basic build pipeline

we saw earlier the basic build pipeline

hey, it works on my machine!



I GUESS ITS TIME TO COMMIT AND PUSH

compile and unit test

integration test

performance test

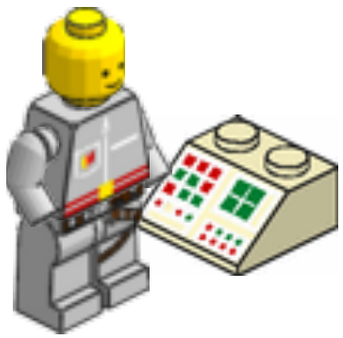
acceptance test

UAT

deploy to production

which environments do you deploy into?

hey, it works on my machine!



compile and unit test

integration test

performance test

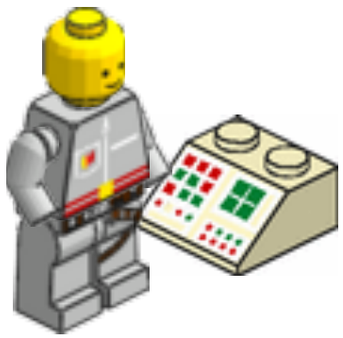
acceptance test

UAT

deploy to production

which environments do you deploy into?

hey, it works on
my machine!



run locally on
build machine

compile and
unit test

integration
test

performance
test

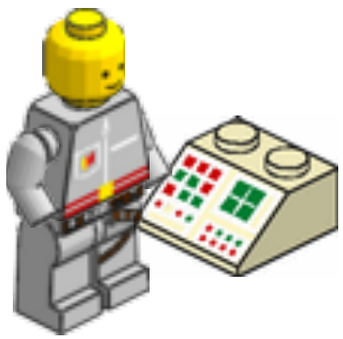
acceptance
test

UAT

deploy to
production

which environments do you deploy into?

hey, it works on my machine!



run locally on build machine

compile and unit test

integration test

performance test

acceptance test

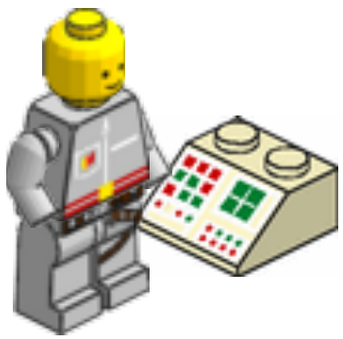
UAT

deploy to production

deployed locally on build machine

which environments do you deploy into?

hey, it works on my machine!



run locally on build machine

compile and unit test

deployed into integration environment

integration test

performance test

acceptance test

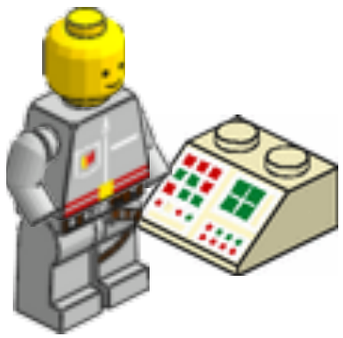
UAT

deploy to production

deployed locally on build machine

which environments do you deploy into?

hey, it works on my machine!



run locally on build machine

compile and unit test

deployed into integration environment

integration test

performance test

deploy to production

acceptance test

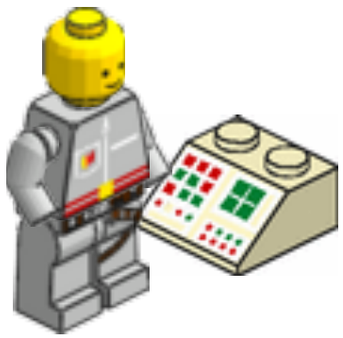
deployed locally on build machine

UAT

deployed into UAT environment

which environments do you deploy into?

hey, it works on my machine!



run locally on build machine

compile and unit test

deployed into integration environment

integration test

deployed into performance environment

performance test

deploy to production

acceptance test

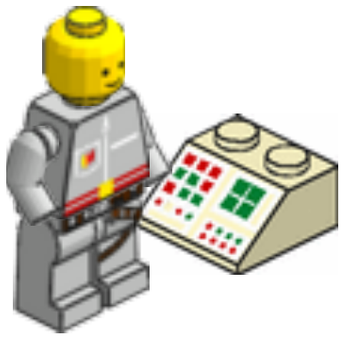
UAT

deployed locally on build machine

deployed into UAT environment

its often only when you start pushing out to bigger environments that you start seeing problems

hey, it works on my machine!



run locally on build machine

compile and unit test

deployed into integration environment

integration test

deployed into performance environment

performance test

deploy to production

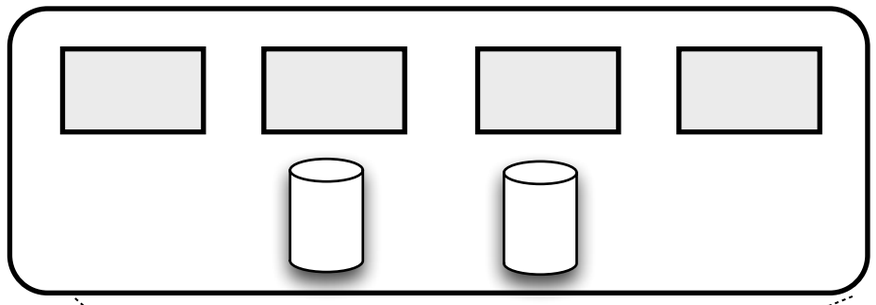
acceptance test

UAT

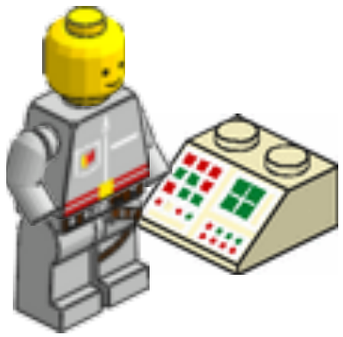
deployed locally on build machine

deployed into UAT environment

its often only when you start pushing out to bigger environments that you start seeing problems



hey, it works on my machine!



run locally on build machine

deployed into integration environment

deployed into performance environment

compile and unit test

integration test

performance test

acceptance test

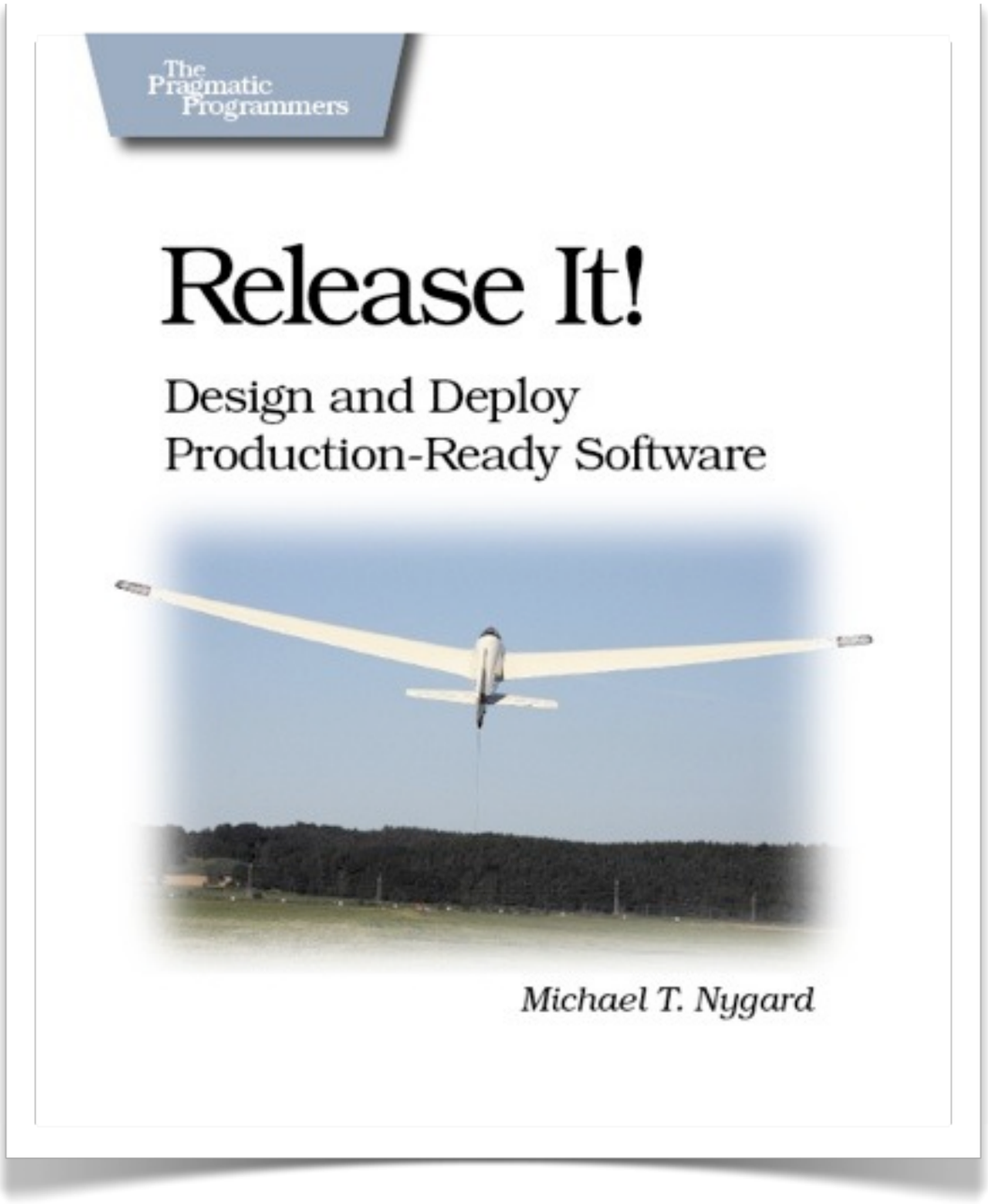
UAT

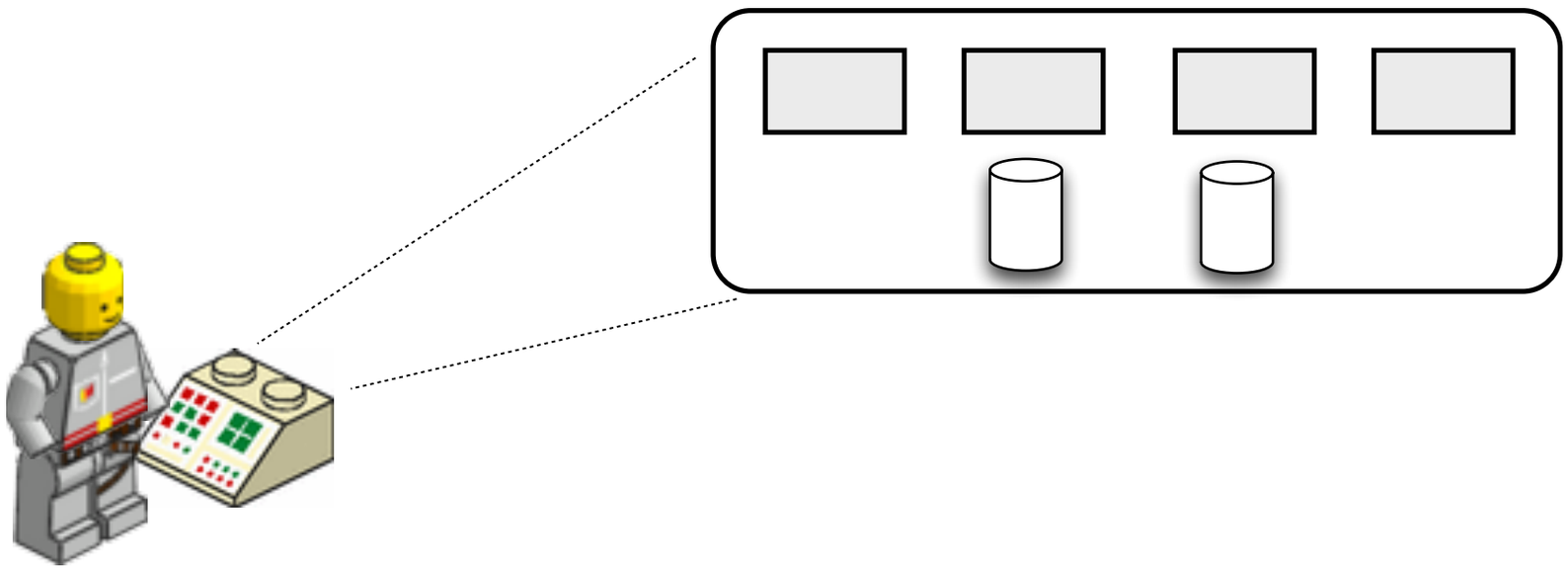
deploy to production

deployed locally on build machine

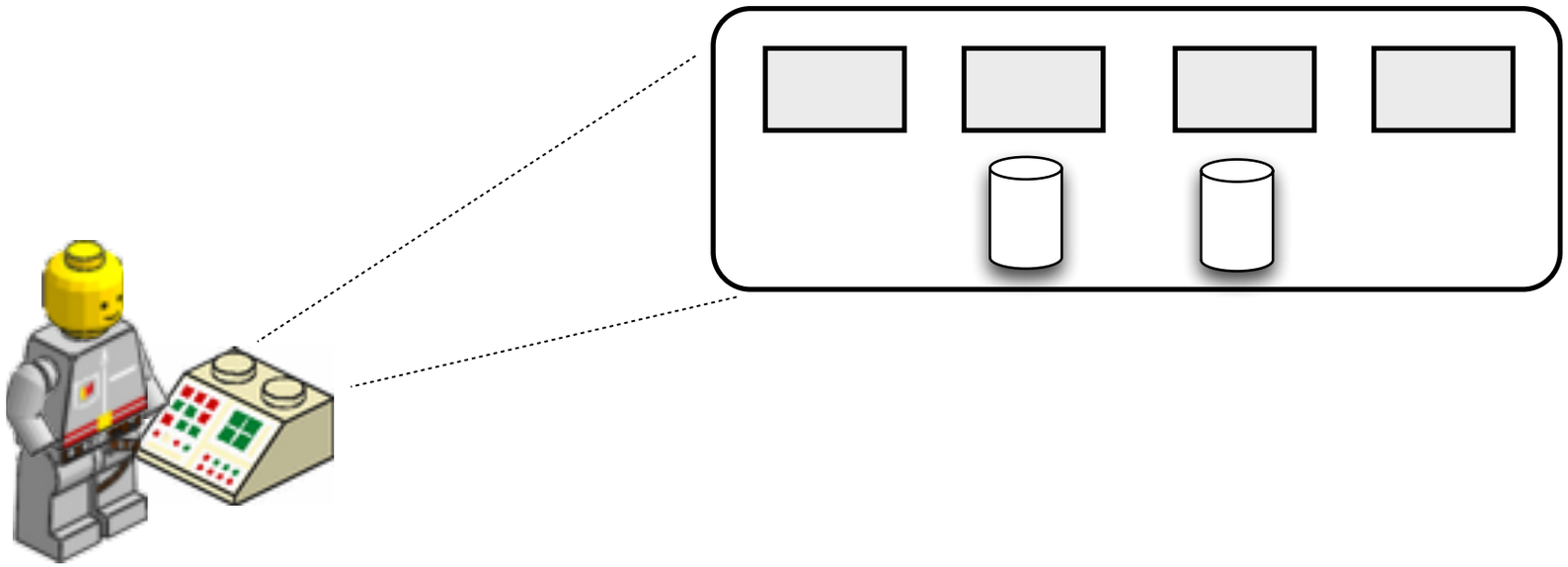
deployed into UAT environment

and if the lovely Mr Nygard taught us anything it was to test with as realistic environments as early as possible



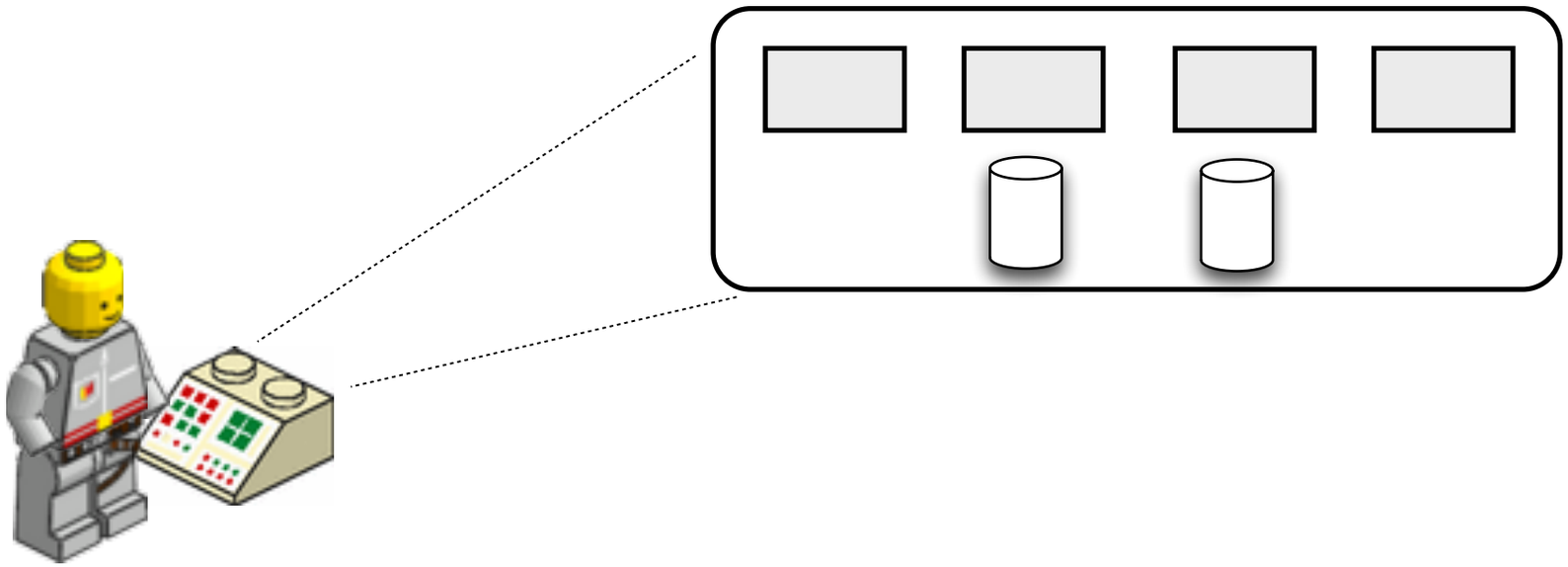


we can use virtualisation on our development machines to bring up whole environments at once



we can use virtualisation on our development machines to bring up whole environments at once

Vagrant



we can use virtualisation on our development machines to bring up whole environments at once

Vagrant

lxc

vagrant looks a bit like this:

vagrant looks a bit like this:

```
$ vagrant box add base http://files.vagrantup.com/lucid32.box
$ vagrant init
$ vagrant up
```

vagrant looks a bit like this:

```
$ vagrant box add base http://files.vagrantup.com/lucid32.box
$ vagrant init
$ vagrant up
```

tough eh? Even I can use it

vagrant looks a bit like this:

```
$ vagrant box add base http://files.vagrantup.com/lucid32.box
$ vagrant init
$ vagrant up
```

tough eh? Even I can use it

under the hood, it applies a puppet manifest (or insert provisioning tool file format)

http://vagrantup.com/v1/docs/getting-started/index.html

multiple VM’s are pretty easy too

multiple VM's are pretty easy too

```
Vagrant::Config.run do |config|
  config.vm.define :web do |web_config|
    web_config.vm.box = "web"
    web_config.vm.forward_port 80, 8080
  end

  config.vm.define :db do |db_config|
    db_config.vm.box = "db"
    db_config.vm.forward_port 3306, 3306
  end
end
```

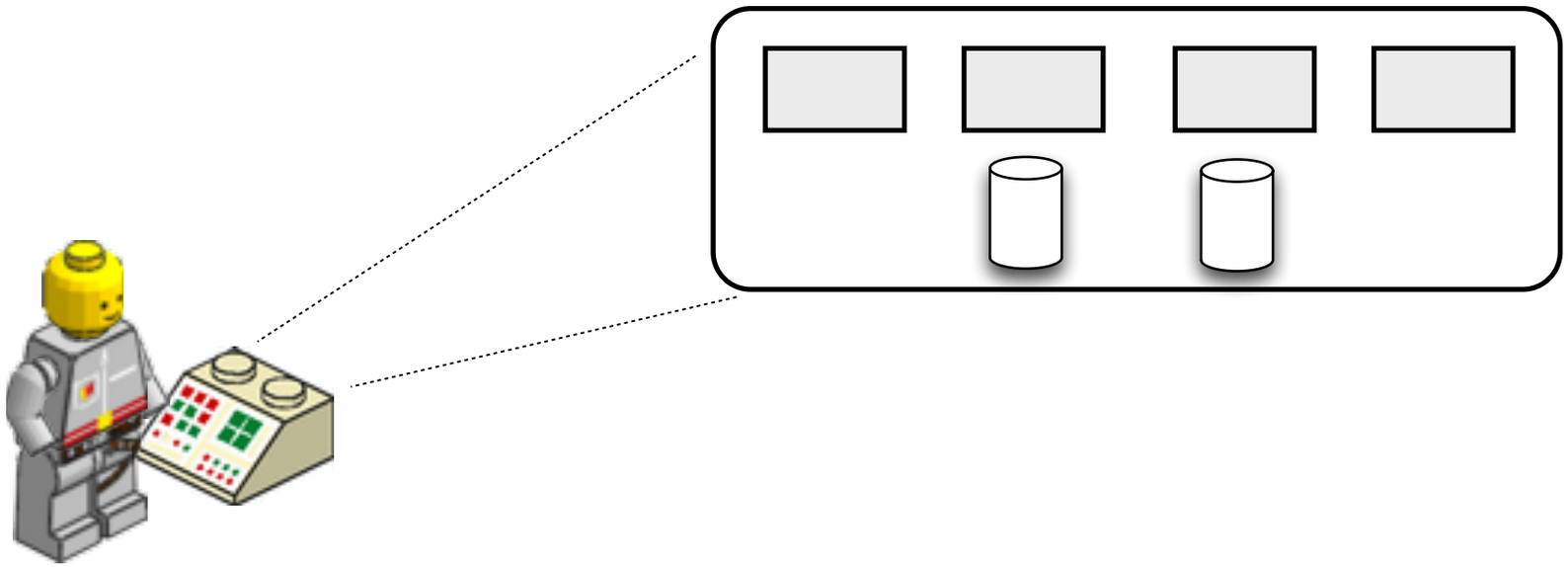
multiple VM's are pretty easy too

```
Vagrant::Config.run do |config|
  config.vm.define :web do |web_config|
    web_config.vm.box = "web"
    web_config.vm.forward_port 80, 8080
  end

  config.vm.define :db do |db_config|
    db_config.vm.box = "db"
    db_config.vm.forward_port 3306, 3306
  end
end
```

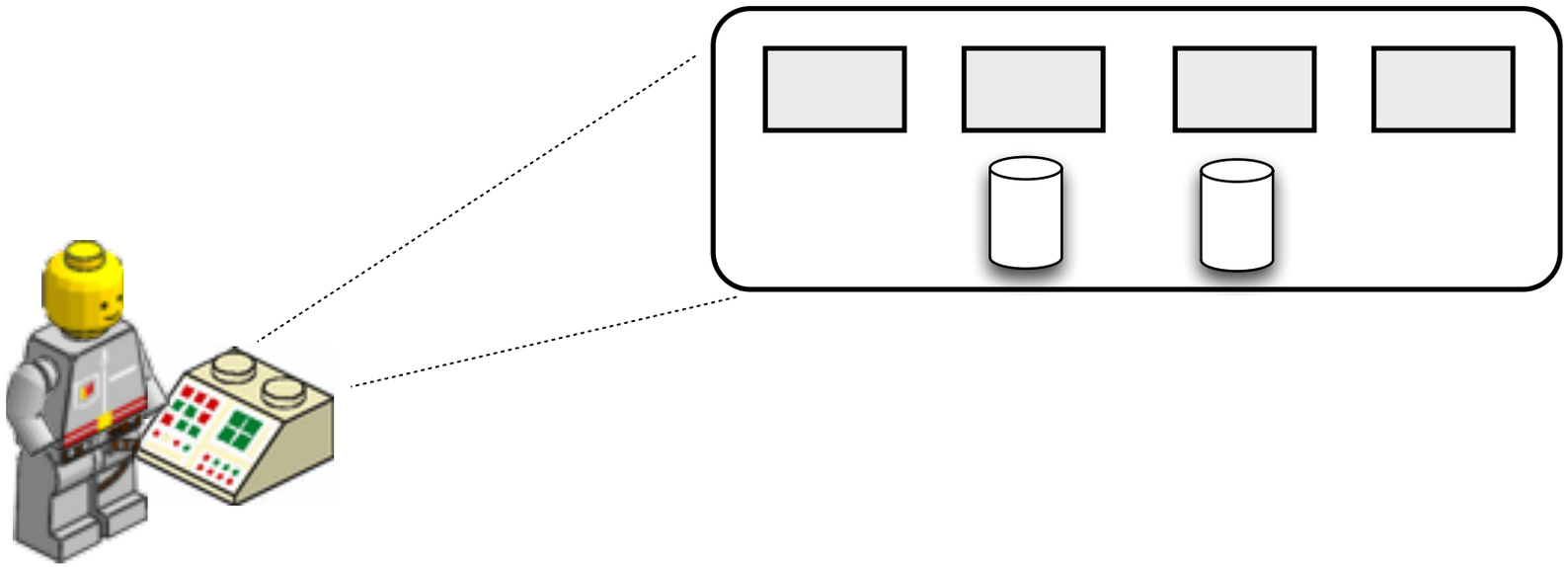
it will take care of the networking business for you

http://vagrantup.com/v1/docs/getting-started/index.html

so now we can deploy our app locally to multiple VM's to test before pushing

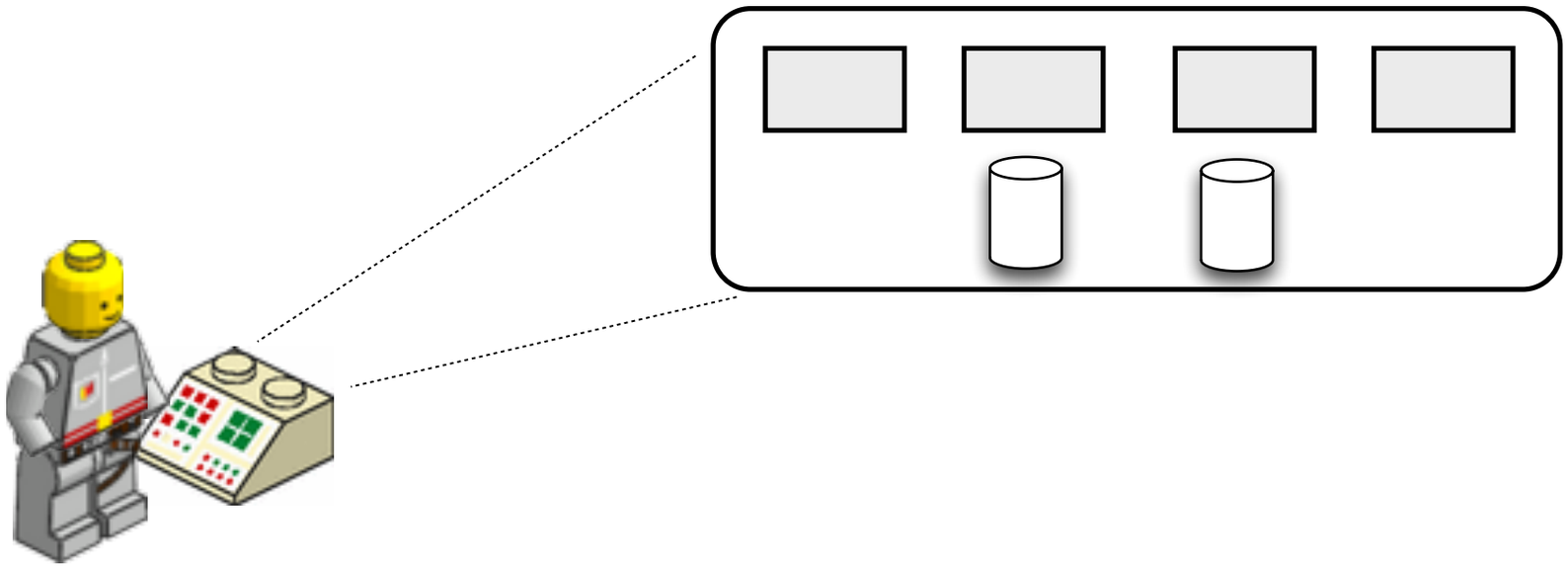


so now we can deploy our app locally to multiple VM's to test before pushing



hey, it really does work on my machine!

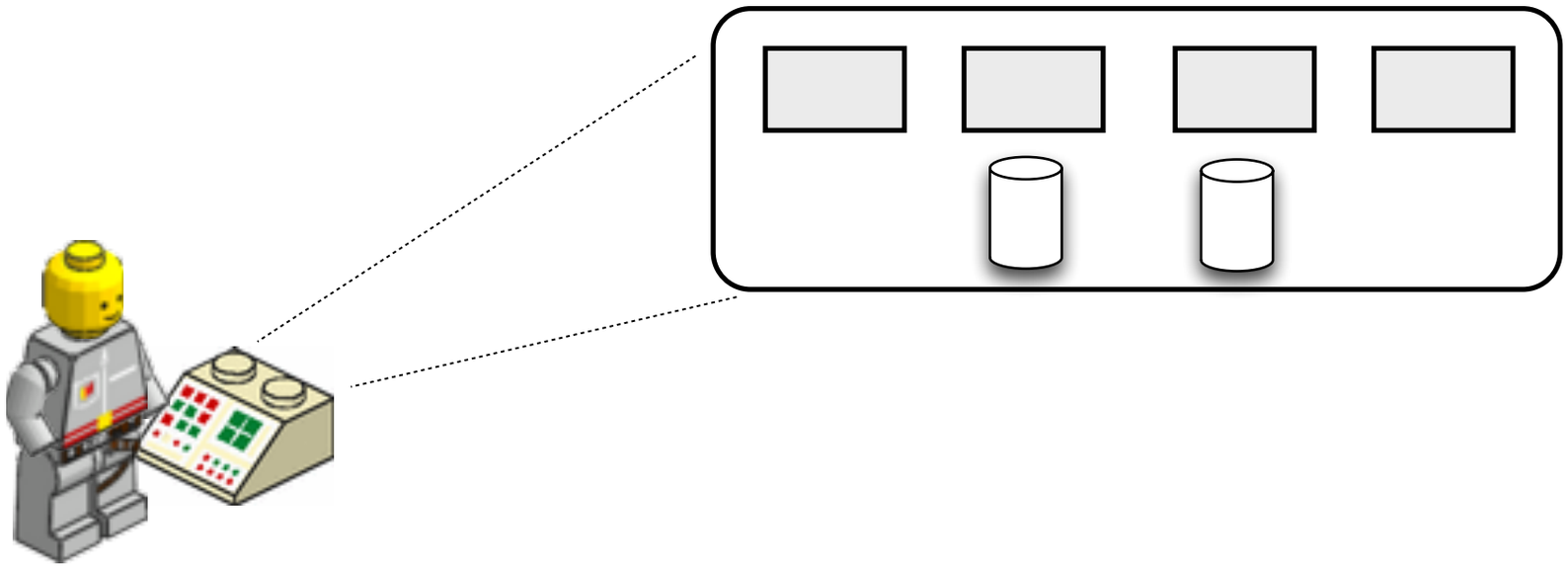
so now we can deploy our app locally to multiple VM's to test before pushing



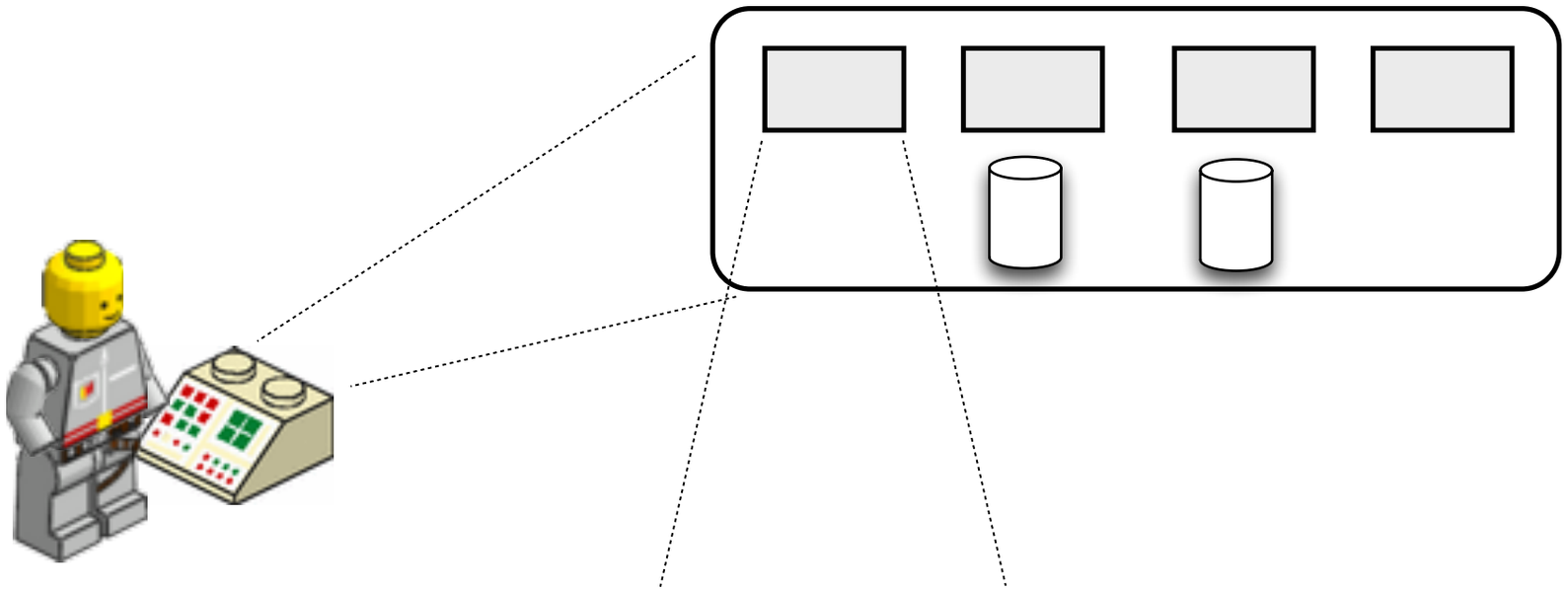
hey, it really does work on my machine!

which is a bit of a win in my book

we specify the dependencies for our application declaratively

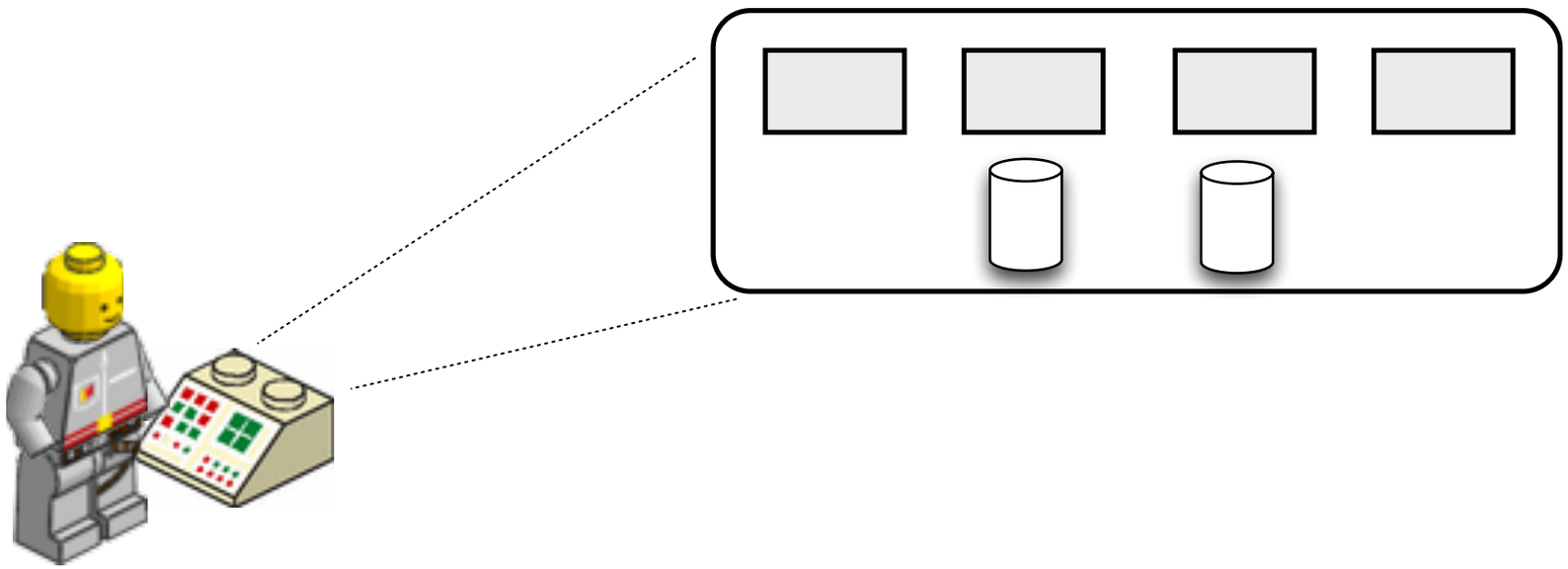


we specify the dependencies for our application declaratively



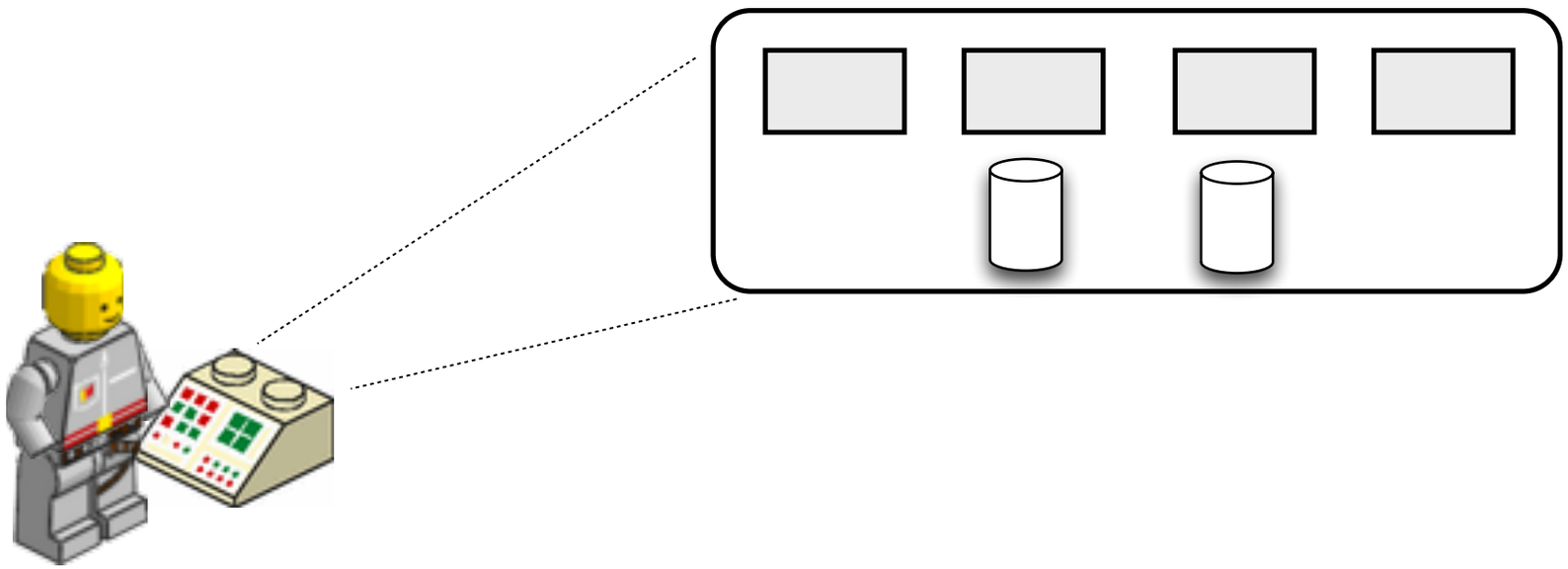
```
class agent( $working_dir, $config_opts ) {
  require platform::sun-java
  require platform::endeavour-service

  File { owner => 0, group => 0, mode => 0644 }

  file { '/etc/motd':
    content => "Welcome to your Agent Node, managed by Puppet.\n"
  }

  platform::service::config {"service_configuration" :
      servicename => "agent",
      servicefilesource => "$working_dir/agent-1.0-SNAPSHOT.jar",
      servicefilename => "agent.jar",
      service_config_opts => $config_opts,
      require => Exec['install-java'];
  }
}
```

and we apply this to each environment we deploy our application to



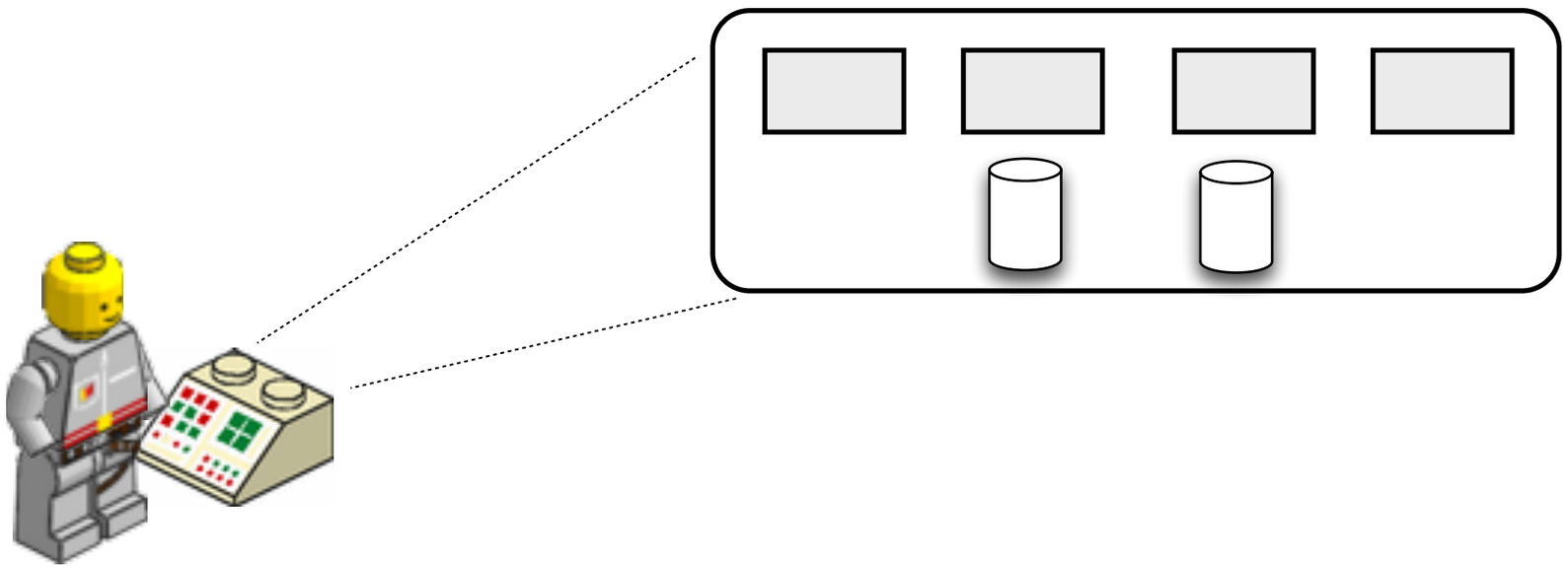
```
class agent( $working_dir, $config_opts ) {
  require platform::sun-java
  require platform::endeavour-service

  File { owner => 0, group => 0, mode => 0644 }

  file { '/etc/motd':
    content => "Welcome to your Agent Node, managed by Puppet.\n"
  }

  platform::service::config {"service_configuration" :
      servicename => "agent",
      servicefilesource => "$working_dir/agent-1.0-SNAPSHOT.jar",
      servicefilename => "agent.jar",
      service_config_opts => $config_opts,
      require => Exec['install-java'];
  }
}
```

and we apply this to each environment we deploy our application to



```
class agent( $working_dir, $config_opts ) {
  require platform::sun-java
  require platform::endeavour-service

  File { owner => 0, group => 0, mode => 0644 }

  file { '/etc/motd':
    content => "Welcome to your Agent Node, managed by Puppet.\n"
  }

  platform::service::config {"service_configuration" :
      servicename => "agent",
      servicefilesource => "$working_dir/agent-1.0-SNAPSHOT.jar",
      servicefilename => 'agent.jar',
      service_config_opts => $config_opts,
      require => Exec['install-java'];
  }
}
```

and we apply this to each environment we deploy our application to



compile and unit test

integration test

performance test

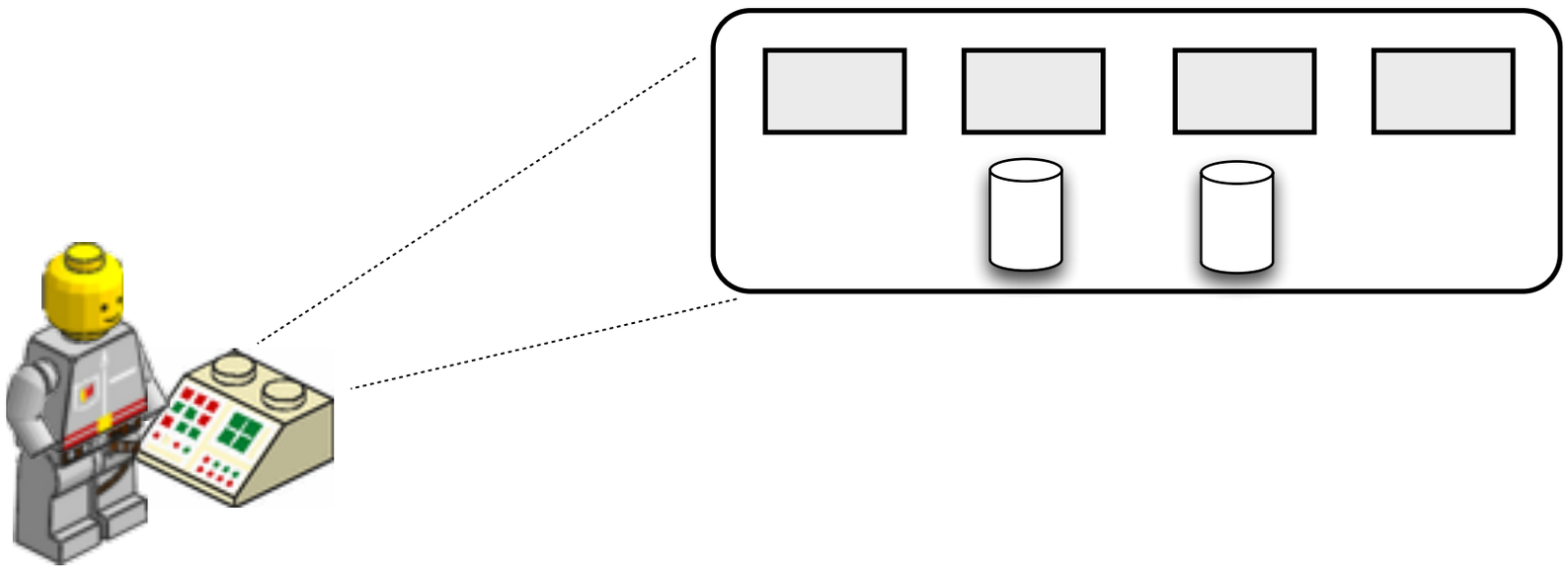
acceptance test

UAT

deploy to production

```
class agent( $working_dir, $config_opts ) {
  require platform::sun-java
  require platform::endeavour-service

  File { owner => 0, group => 0, mode => 0644 }

  file { '/etc/motd':
    content => "Welcome to your Agent Node, managed by Puppet.\n"
  }

  platform::service::config {"service_configuration" :
      servicename => "agent",
      servicefilesource => "$working_dir/agent-1.0-SNAPSHOT.jar",
      servicefilename => 'agent.jar',
      service_config_opts => $config_opts,
      require => Exec['install-java'];
  }
}
```

and we apply this to each environment we deploy our application to



compile and unit test

integration test

performance test

acceptance test

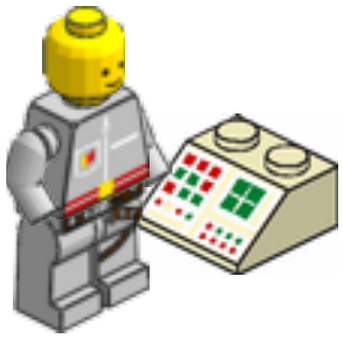
UAT

deploy to production

```
class agent( $working_dir, $config_opts ) {
  require platform::sun-java
  require platform::endeavour-service

  File { owner => 0, group => 0, mode => 0644 }

  file { '/etc/motd':
    content => "Welcome to your Agent Node, managed by Puppet.\n"
  }

  platform::service::config {"service_configuration" :
      servicename => "agent",
      servicefilesource => "$working_dir/agent-1.0-SNAPSHOT.jar",
      servicefilename => "agent.jar",
      service_config_opts => $config_opts,
      require => Exec['install-java'];
  }
}
```

and we abstract away the fact that
we are pushing to a local Data Centre
or to an IaaS platform like AWS

you may be using an IaaS provider to support testing and integration but deploy into a local production environment



run locally on build machine

integration environment (IaaS)

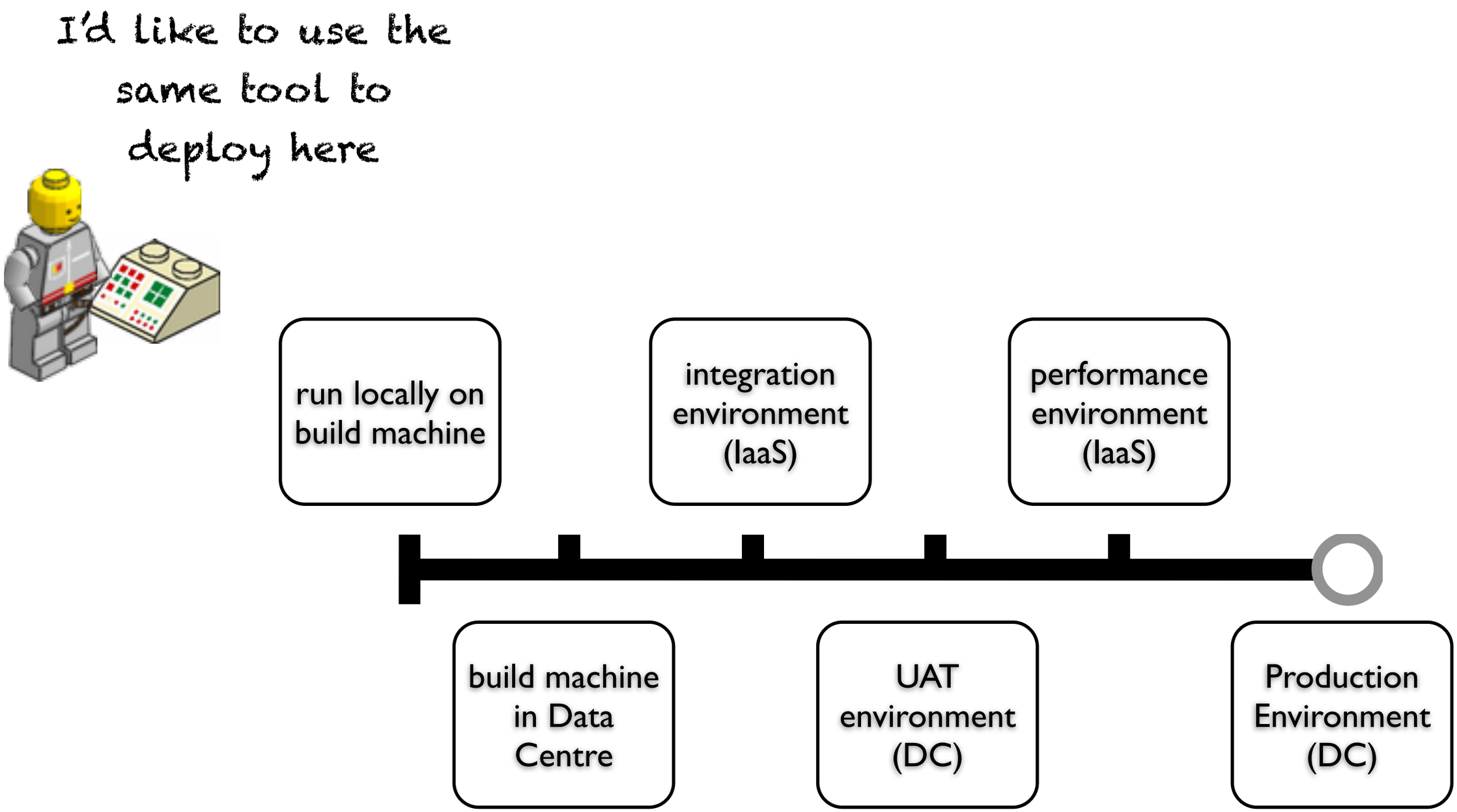
performance environment (IaaS)

build machine in Data Centre

UAT environment (DC)

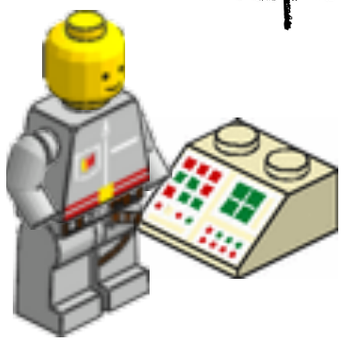
Production Environment (DC)

you may be using an IaaS provider to support testing and integration but deploy into a local production environment



you may be using an IaaS provider to support testing and integration but deploy into a local production environment

I'd like to use the same tool to deploy here



here

run locally on build machine

integration environment (IaaS)

performance environment (IaaS)

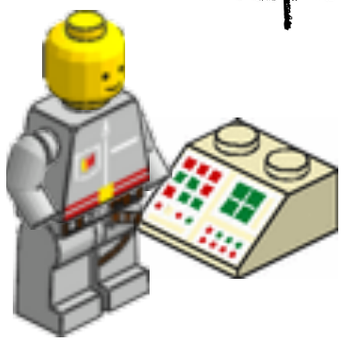
build machine in Data Centre

UAT environment (DC)

Production Environment (DC)

you may be using an IaaS provider to support testing and integration but deploy into a local production environment

I'd like to use the same tool to deploy here



here

run locally on build machine

integration environment (IaaS)

performance environment (IaaS)

build machine in Data Centre

UAT environment (DC)

Production Environment (DC)

and here

and there is tooling available to do this too

and there is tooling available to do this too

```
@task
def deploy_environment(environment, build_number):
    target_env = ENVIRONMENTS[environment]
    deployment = Deployment(build_number,target_env)
    deployment.install(SERVICES)

@task
def end_to_end_test_against(environment, build_number):
    target_env = ENVIRONMENTS[environment]
    deployment = Deployment(build_number,target_env)
    deployment.run_end_to_end_tests(EndToEndTest())
```

and there is tooling available to do this too

```
@task
def deploy_environment(environment, build_number):
    target_env = ENVIRONMENTS[environment]
    deployment = Deployment(build_number,target_env)
    deployment.install(SERVICES)

@task
def end_to_end_test_against(environment, build_number):
    target_env = ENVIRONMENTS[environment]
    deployment = Deployment(build_number,target_env)
    deployment.run_end_to_end_tests(EndToEndTest())
```

this is a tool called Fabric

it’s a wrapper over SSH so you can run commands against anything you have SSH access to

it’s a wrapper over SSH so you can run commands against anything you have SSH access to

```
class RemoteServer(ApplicationServer):

    def deploy_service(self, target_env, service_name, artefact_repo):
          with settings(user='service', host_string=self.host, key_filename='~/.ssh/
deploy'):
            run('rm -rf /tmp/deploy.%s' % artefact_repo.build_number)
            run('svn checkout /trunk/platform_deploy /tmp/deploy.%s' %
                artefact_repo.build_number
            )
            run('/tmp/deploy.%s/deploy-no-db.sh %s %s %s %s' % (
                artefact_repo.build_number,
                service_name,
                artefact_repo.build_number,
                self.install_dir,
                target_env.property_file_name)
            )
```

```
class LocalServer(ApplicationServer):

    def deploy_service(self, target_env, service_name, artefact_repo):
        local('rm -rf /tmp/deploy.%s' % artefact_repo.build_number)
        local('svn checkout /trunk/platform_deploy /tmp/deploy.%s' %
            artefact_repo.build_number
        )
        local('/tmp/deploy.%s/deploy-no-db.sh %s %s %s %s' % (
            artefact_repo.build_number,
            service_name,
            artefact_repo.build_number,
            self.install_dir,
            target_env.property_file_name)
        )
```

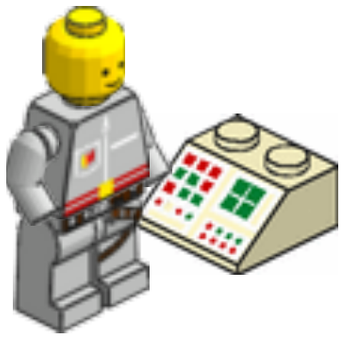
```
class RemoteServer(ApplicationServer):

    def deploy_service(self, target_env, service_name, artefact_repo):
          with settings(user='service', host_string=self.host, key_filename='~/.ssh/
deploy'):
            run('rm -rf /tmp/deploy.%s' % artefact_repo.build_number)
            run('svn checkout /trunk/platform_deploy /tmp/deploy.%s' %
                artefact_repo.build_number
            )
            run('/tmp/deploy.%s/deploy-no-db.sh %s %s %s %s' % (
                artefact_repo.build_number,
                service_name,
                artefact_repo.build_number,
                self.install_dir,
                target_env.property_file_name)
            )
```

and yes, the astute among you will have noticed the awful duplication

```
class LocalServer(ApplicationServer):

    def deploy_service(self, target_env, service_name, artefact_repo):
        local('rm -rf /tmp/deploy.%s' % artefact_repo.build_number)
        local('svn checkout /trunk/platform_deploy /tmp/deploy.%s' %
            artefact_repo.build_number
        )
        local('/tmp/deploy.%s/deploy-no-db.sh %s %s %s %s' % (
            artefact_repo.build_number,
            service_name,
            artefact_repo.build_number,
            self.install_dir,
            target_env.property_file_name)
        )
```

```
class RemoteServer(ApplicationServer):

    def deploy_service(self, target_env, service_name, artefact_repo):
          with settings(user='service', host_string=self.host, key_filename='~/.ssh/
deploy'):
            run('rm -rf /tmp/deploy.%s' % artefact_repo.build_number)
            run('svn checkout /trunk/platform_deploy /tmp/deploy.%s' %
                artefact_repo.build_number
            )
            run('/tmp/deploy.%s/deploy-no-db.sh %s %s %s %s' % (
                artefact_repo.build_number,
                service_name,
                artefact_repo.build_number,
                self.install_dir,
                target_env.property_file_name)
            )
```

```
$ fab deploy_environment:integration,build-789
```



run locally on build machine

integration environment (IaaS)

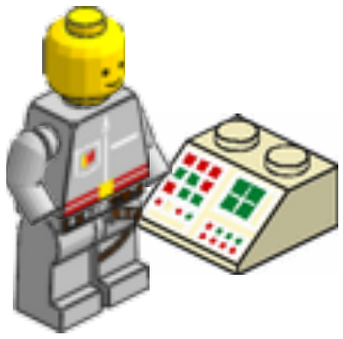
performance environment (IaaS)

build machine in Data Centre

UAT environment (DC)

Production Environment (DC)

```
$ fab deploy_environment:integration,build-789
```



so I can use the same command line here

run locally on build machine

integration environment (IaaS)

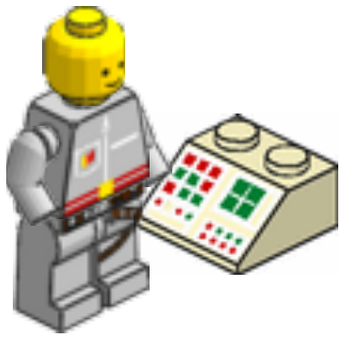
performance environment (IaaS)

build machine in Data Centre

UAT environment (DC)

Production Environment (DC)

```
$ fab deploy_environment:integration,build-789
```



so I can use the same command line here

as the CD server does to push here

run locally on build machine

integration environment (IaaS)

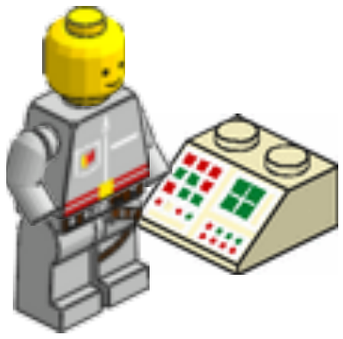
performance environment (IaaS)

build machine in Data Centre

UAT environment (DC)

Production Environment (DC)

```
$ fab deploy_environment:integration,build-789
```



so I can use the same command line here

as the CD server does to push here

run locally on build machine

integration environment (IaaS)

performance environment (IaaS)

build machine in Data Centre

UAT environment (DC)

Production Environment (DC)

or here

declarative environment provisioning

```
prod:
  nodes:
  - ami_id: ami-4dad7424
    size:   t1.micro
    credentials_name: us-east-ssh
    aws_key_name : test
    services: [hello_world]
    security_groups: [ spicy-beef ]
    availability_zone: us-east-1a
    type: phoenix.providers.aws_provider.AWSNodeDefinition
  - ami_id: ami-4dad7424
    size:   t1.micro
    credentials_name: us-east-ssh
    aws_key_name : test
    services: [hello_world]
    security_groups: [ spicy-beef ]
    availability_zone: us-east-1b
    type: phoenix.providers.aws_provider.AWSNodeDefinition
  - ami_id: ami-4dad7424
    size:   t1.micro
    credentials_name: us-east-ssh
    aws_key_name : test
    services: [apache]
    type: phoenix.providers.aws_provider.AWSNodeDefinition
    security_groups: [ spicy-beef ]

  node_provider:
    class_name: AWSNodeProvider
    public_api_key: {{ aws_public_api_key }}
    private_api_key: {{ aws_private_api_key }}
```

declarative environment provisioning

```
prod:
  nodes:
  - ami_id: ami-4dad7424
    size:   t1.micro
    credentials_name: us-east-
    aws_key_name : test
    services: [hello_world]
    security_groups: [ spicy-h
    availability_zone: us-east
    type: phoenix.providers.av
  - ami_id: ami-4dad7424
    size:   t1.micro
    credentials_name: us-east-
    aws_key_name : test
    services: [hello_world]
    security_groups: [ spicy-h
    availability_zone: us-east
    type: phoenix.providers.av
- ami_id: ami-4dad7424
    size:   t1.micro
    credentials_name: us-east-
    aws_key_name : test
    services: [apache]
    type: phoenix.providers.av
    security_groups: [ spicy-h

  node_provider:
    class_name: AWSNodeProvide
    public_api_key: {{ aws_pub
    private_api_key: {{ aws_pr
```

```
apache:
  puppet_module_directory : puppet
  puppet_manifest : apache.pp
  service_configurator:
phoenix.configurators.puppet_service_configurator.PuppetServiceConfigurator
  connectivity:
    - protocol: tcp
      ports: [ 80 ]
      allowed: [ WORLD ]

hello_world:
  puppet_module_directory : puppet
  puppet_manifest : hello_world.pp
  service_configurator:
phoenix.configurators.puppet_service_configurator.PuppetServiceConfigurator
  connectivity:
    - protocol: tcp
      ports: [ 8080, 8081 ]
      allowed: [ WORLD ]

mongo:
  puppet_module_directory : puppet
  puppet_manifest : mongo.pp
  service_configurator:
phoenix.configurators.puppet_service_configurator.PuppetServiceConfigurator
  connectivity:
    - protocol: tcp
      ports: [ 27017 ]
      allowed: [ hello_world ]
```

declarative environment provisioning

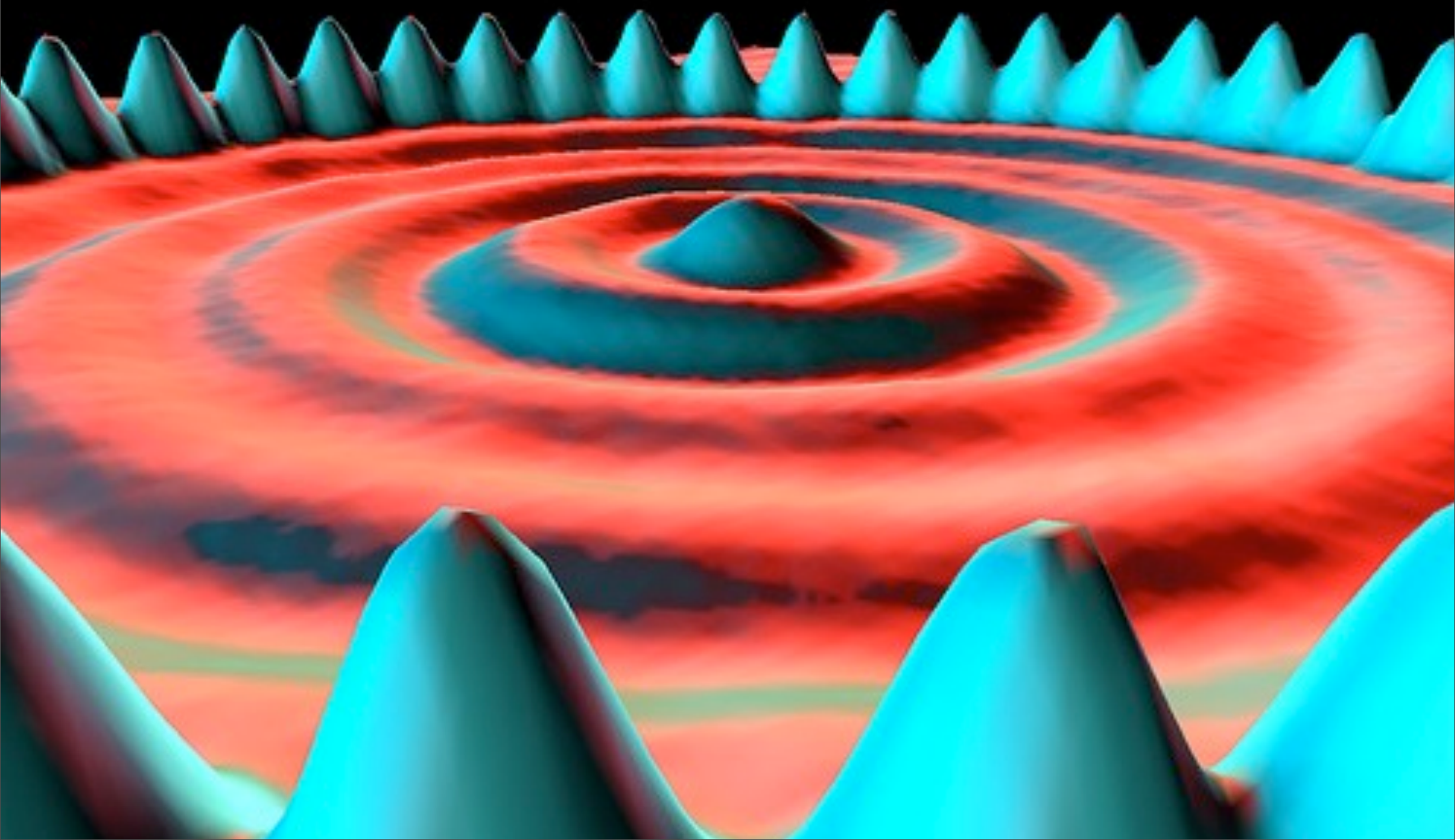
```
prod:
  nodes:
  - ami_id: ami-4dad7424
    size:   t1.micro
    credentials_name: us-east-ssh
    aws_key_name : test
    services: [hello_world]
    security_groups: [ spicy-beef ]
    availability_zone: us-east-1b
    type: phoenix.providers.aws_provider.AWSNodeDefinition
  - ami_id: ami-4dad7424
    size:   t1.micro
    credentials_name: us-east-ssh
    aws_key_name : test
    services: [apache]
    type: phoenix.providers.aws_provider.AWSNodeDefinition
    security_groups: [ spicy-beef ]

  node_provider:
    class_name: AWSNodeProvider
    public_api_key: {{ aws_public_api_key }}
    private_api_key: {{ aws_private_api_key }}
```

OpsWorks

```
apache:
  puppet_module_directory : puppet
  puppet_manifest : apache.pp
  service_configurator:
phoenix.configurators.puppet_service_configurator.PuppetServiceConfigurator
  connectivity:
    - protocol: tcp
      ports: [ 80 ]
      allowed: [ WORLD ]

hello_world:
  puppet_module_directory : puppet
  puppet_manifest : hello_world.pp
  service_configurator:
phoenix.configurators.puppet_service_configurator.PuppetServiceConfigurator
  connectivity:
    - protocol: tcp
      ports: [ 8080, 8081 ]
      allowed: [ WORLD ]

mongo:
  puppet_module_directory : puppet
  puppet_manifest : mongo.pp
  service_configurator:
phoenix.configurators.puppet_service_configurator.PuppetServiceConfigurator
  connectivity:
    - protocol: tcp
      ports: [ 27017 ]
      allowed: [ hello_world ]
```

Checkmate

Deployment



the Fowler bomb

the Fowler bomb

“it pushes the accidental complexity into the infrastructure”

Martin Fowler

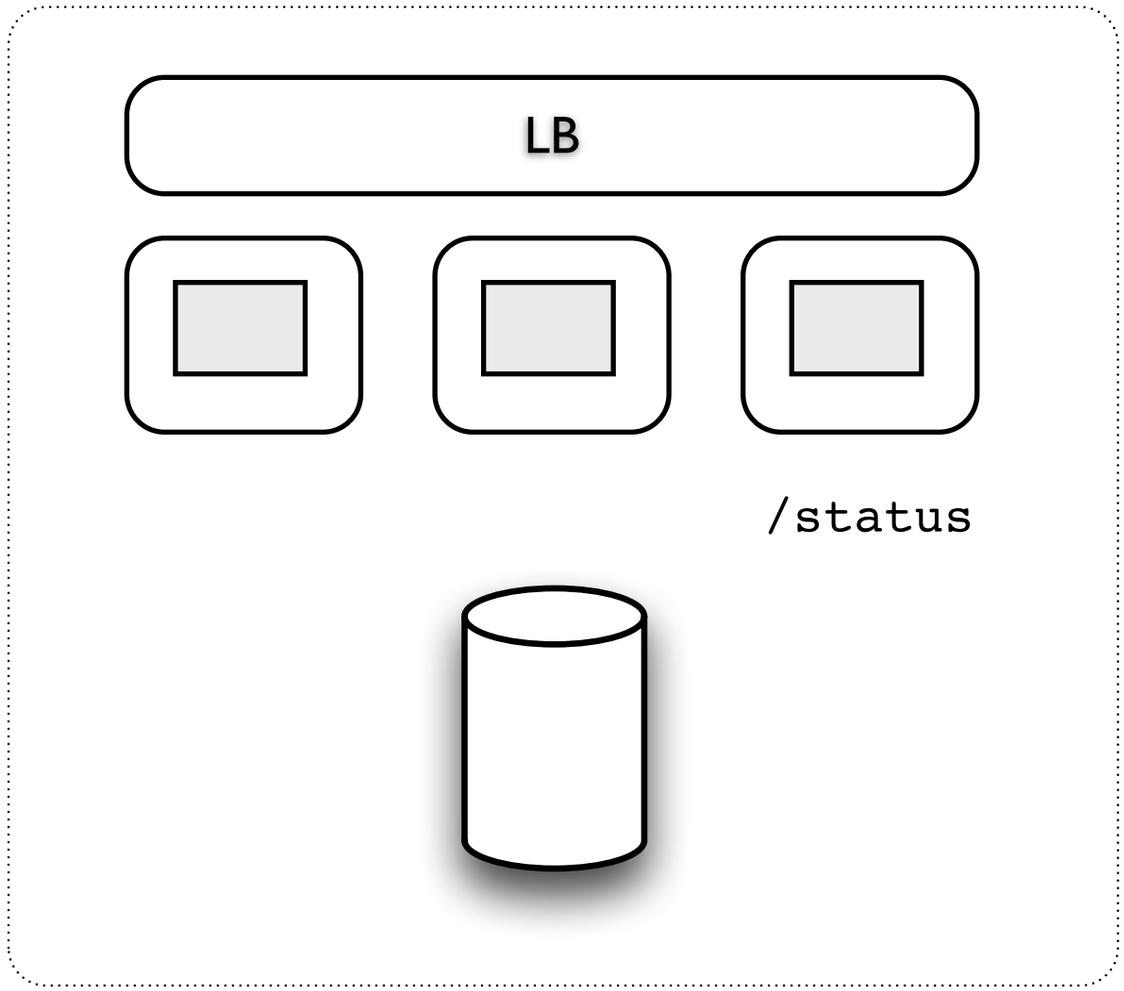
the Fowler bomb

“it pushes the accidental complexity into the infrastructure”

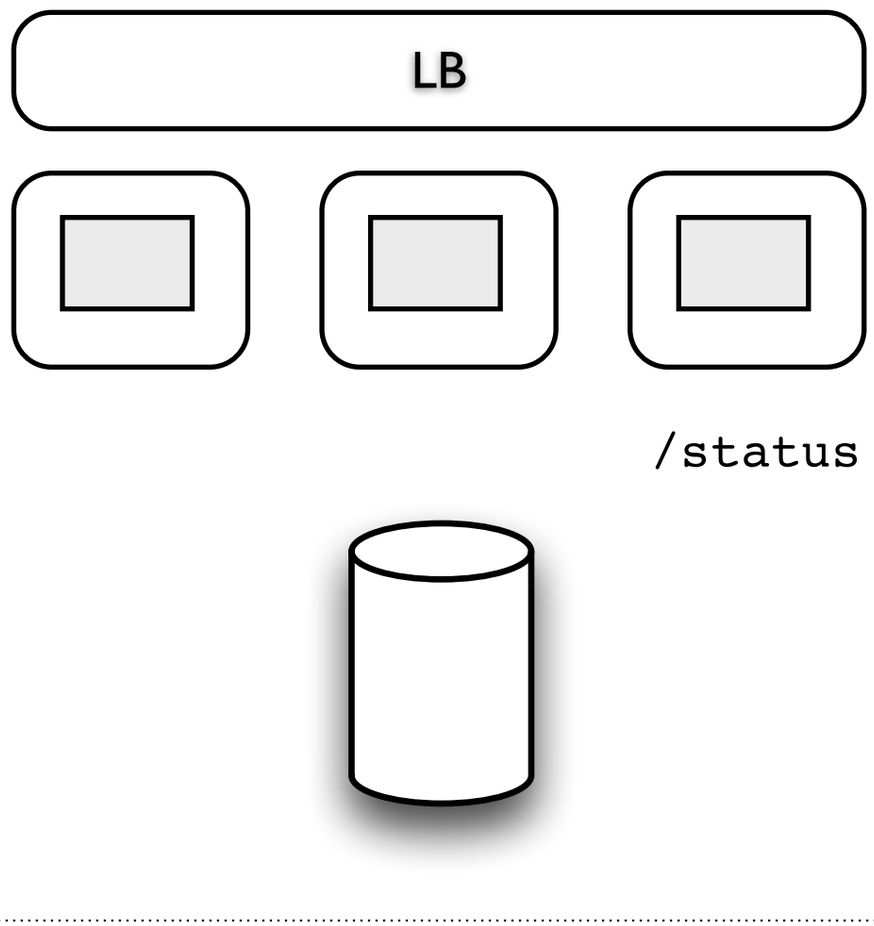
Martin Fowler

turns out that monitoring and logging have a big part to play too

monitoring needs to be a first class concern

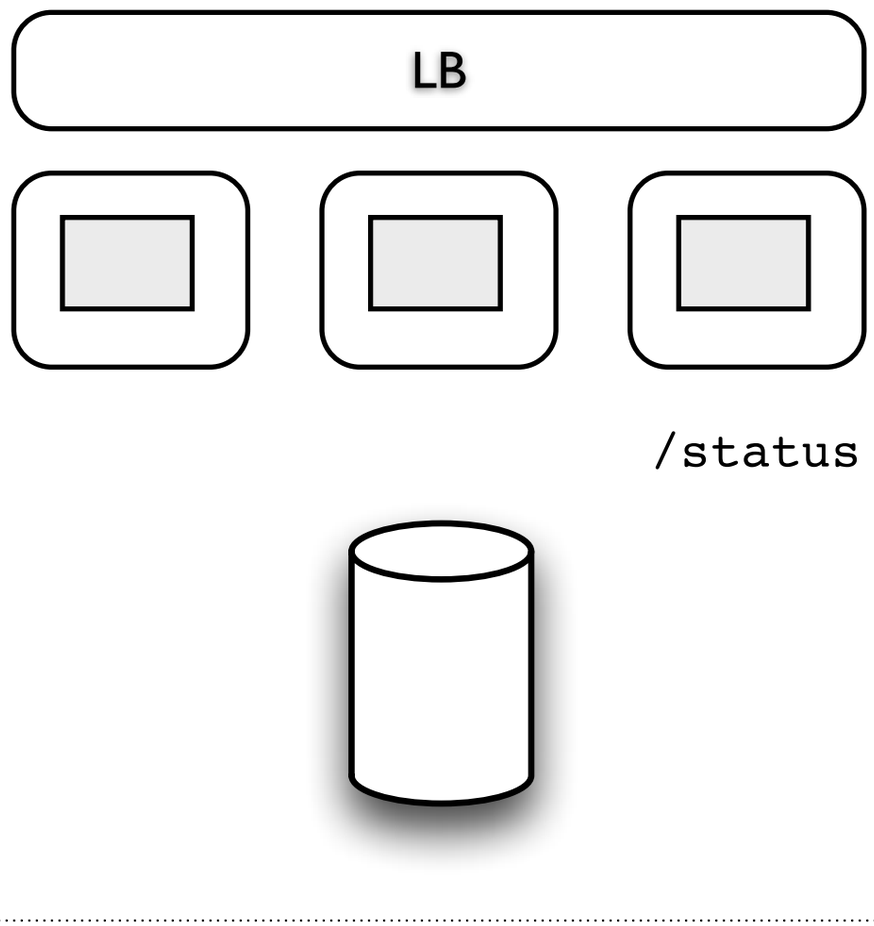


monitoring needs to be a first class concern



each of these should report metrics
at a well known location

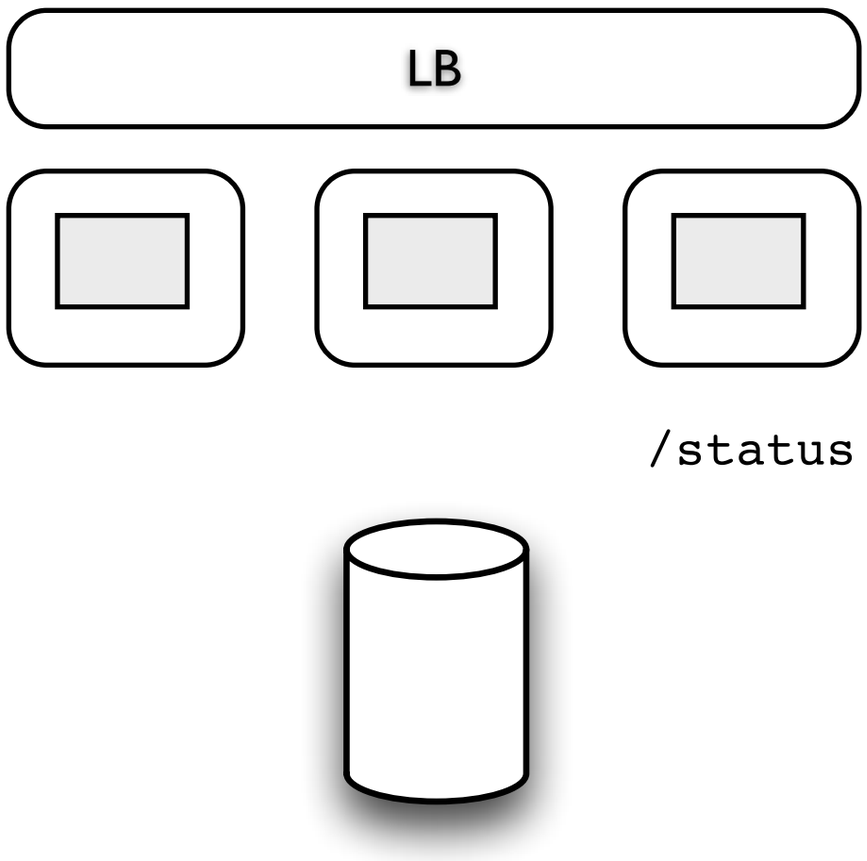
monitoring needs to be a first class concern



each of these should report metrics at a well known location

these metrics can displayed locally

monitoring needs to be a first class concern

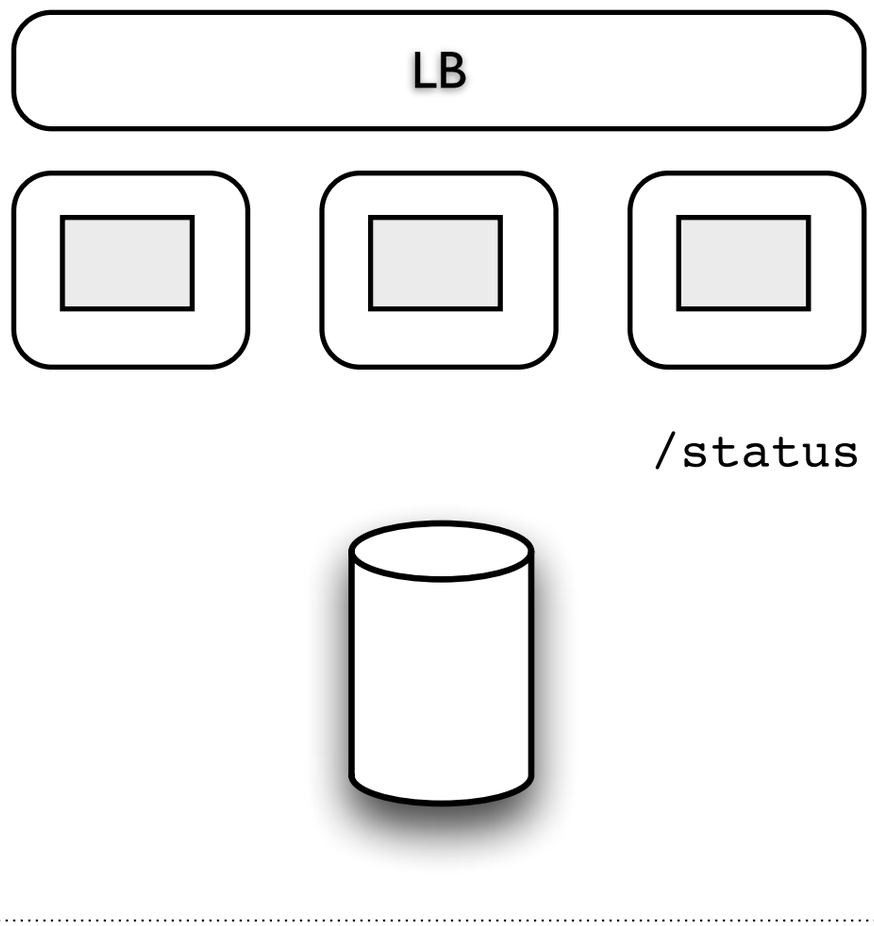


each of these should report metrics at a well known location

these metrics can displayed locally

and pushed to centralised monitoring tools

monitoring needs to be a first class concern



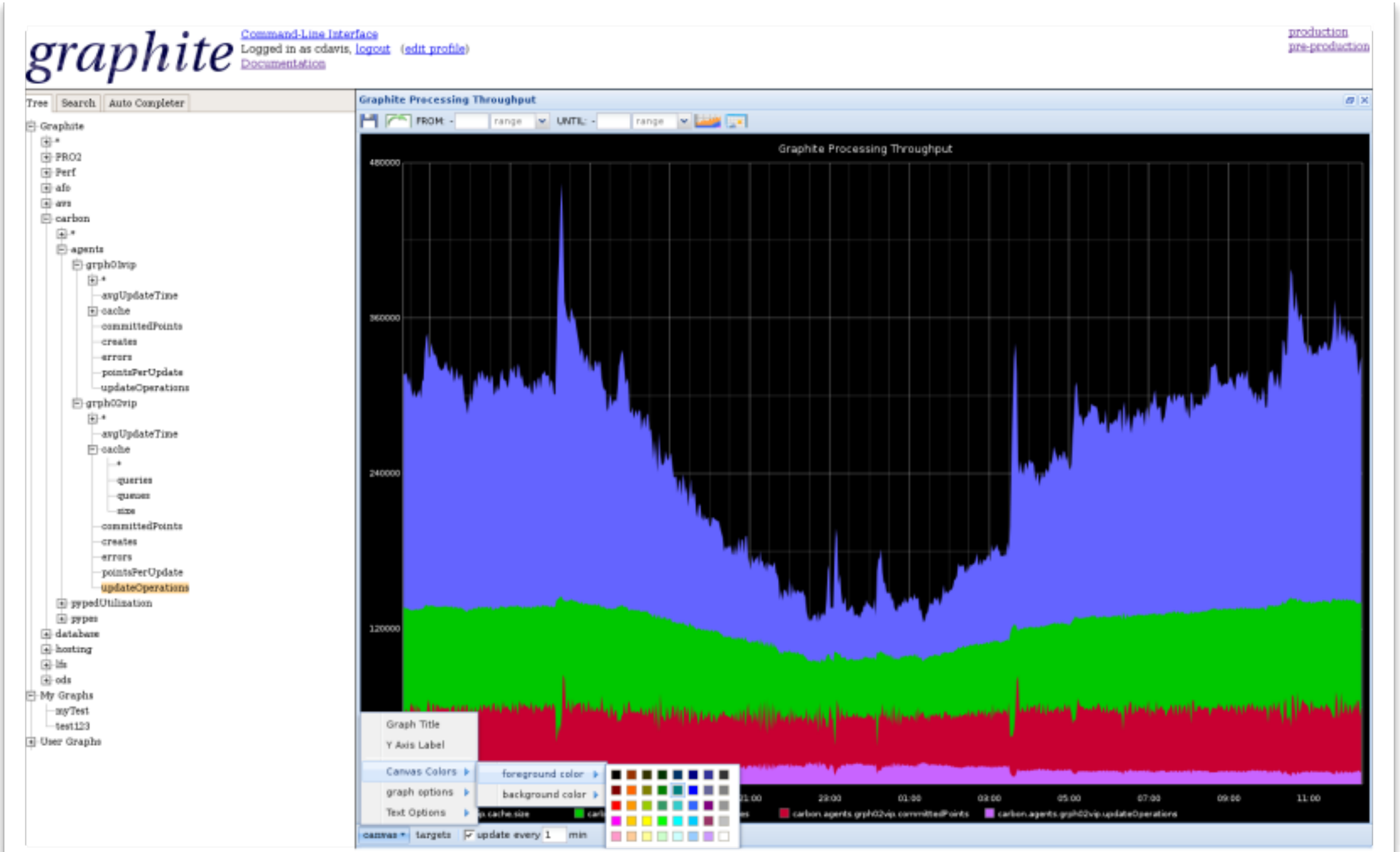
each of these should report metrics at a well known location

these metrics can displayed locally

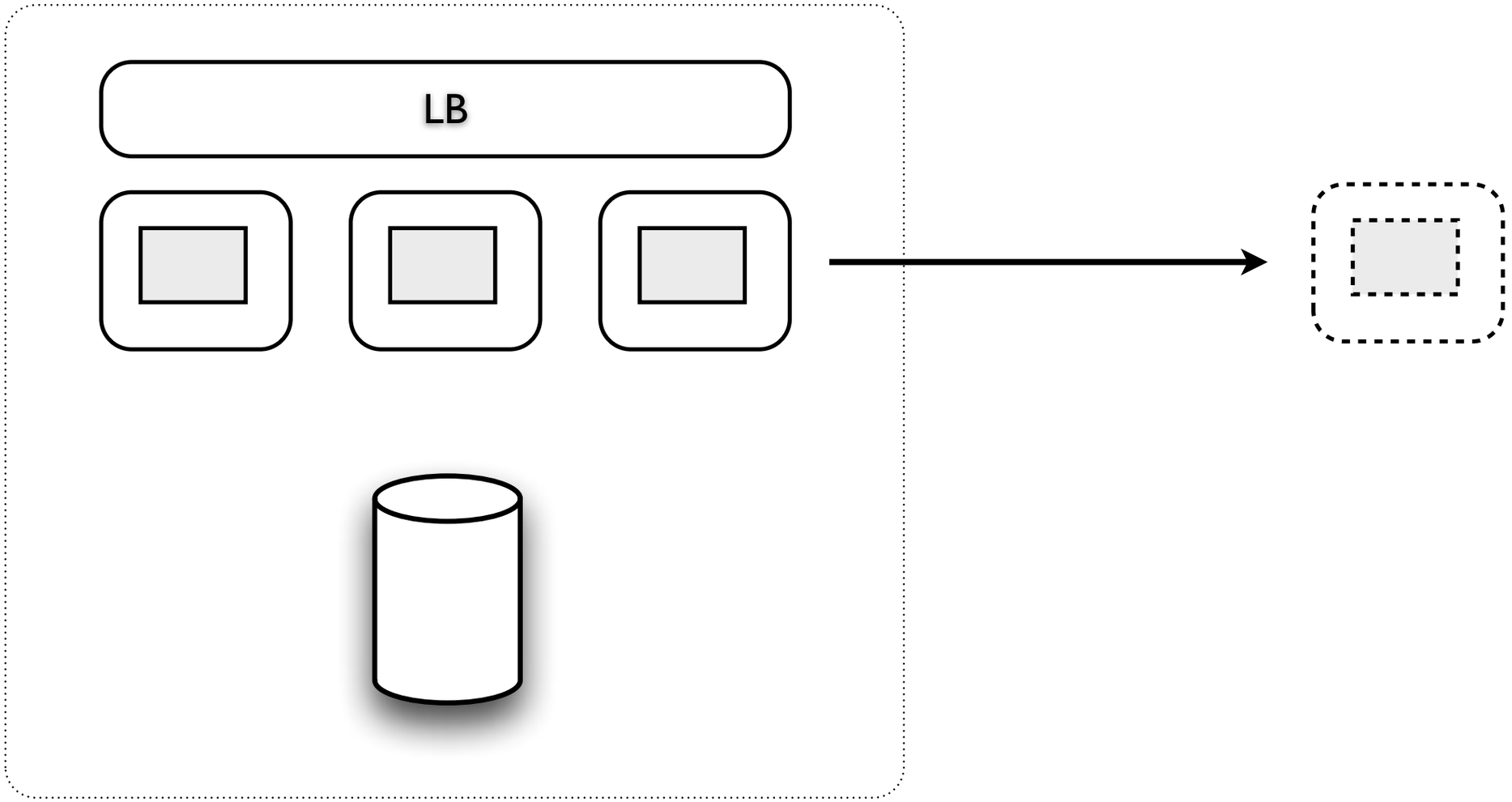
and pushed to centralised monitoring tools

Status aware applications are the way forward

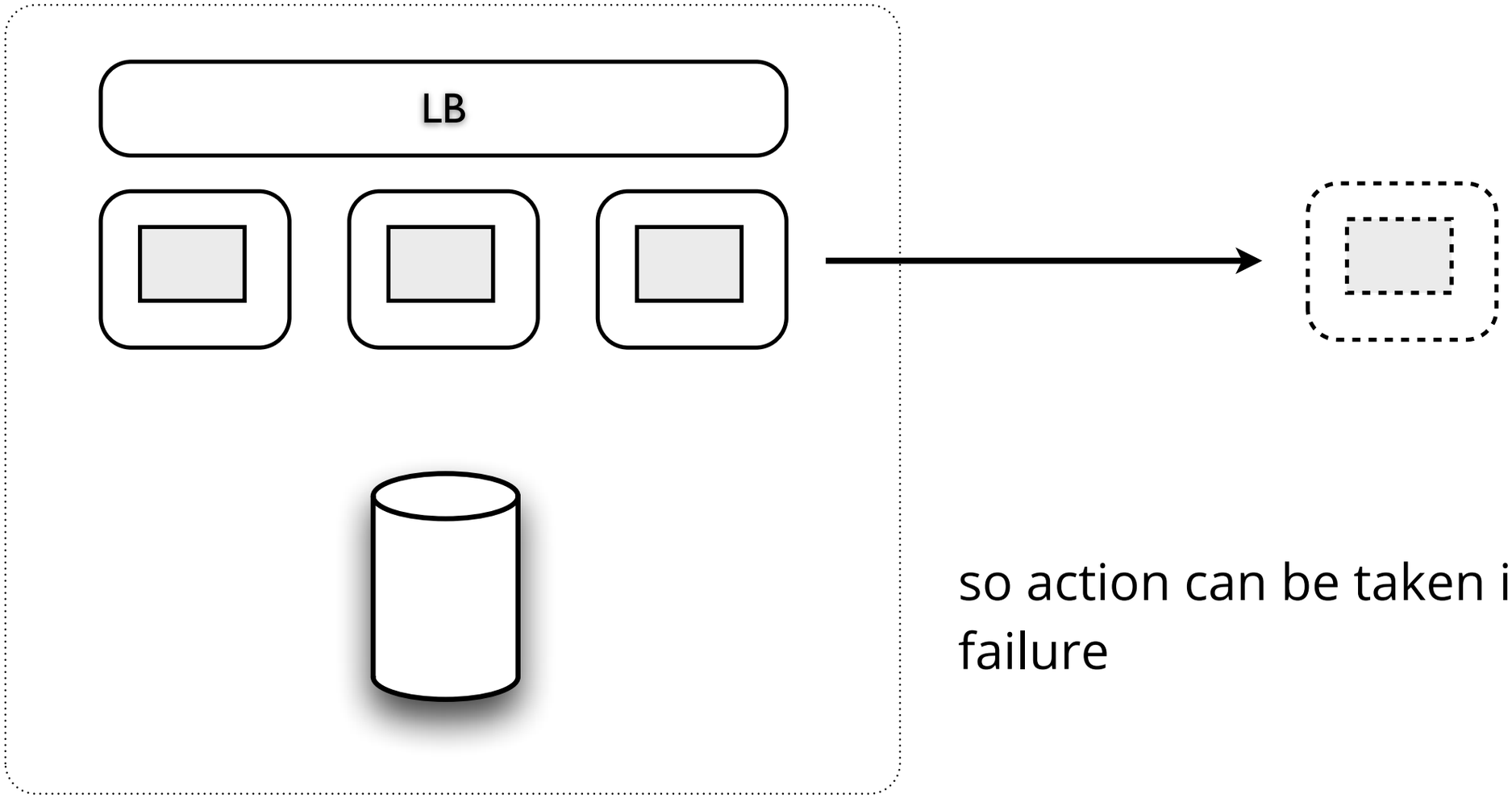
graphite for example



downstream systems and integration points should be monitored by each service

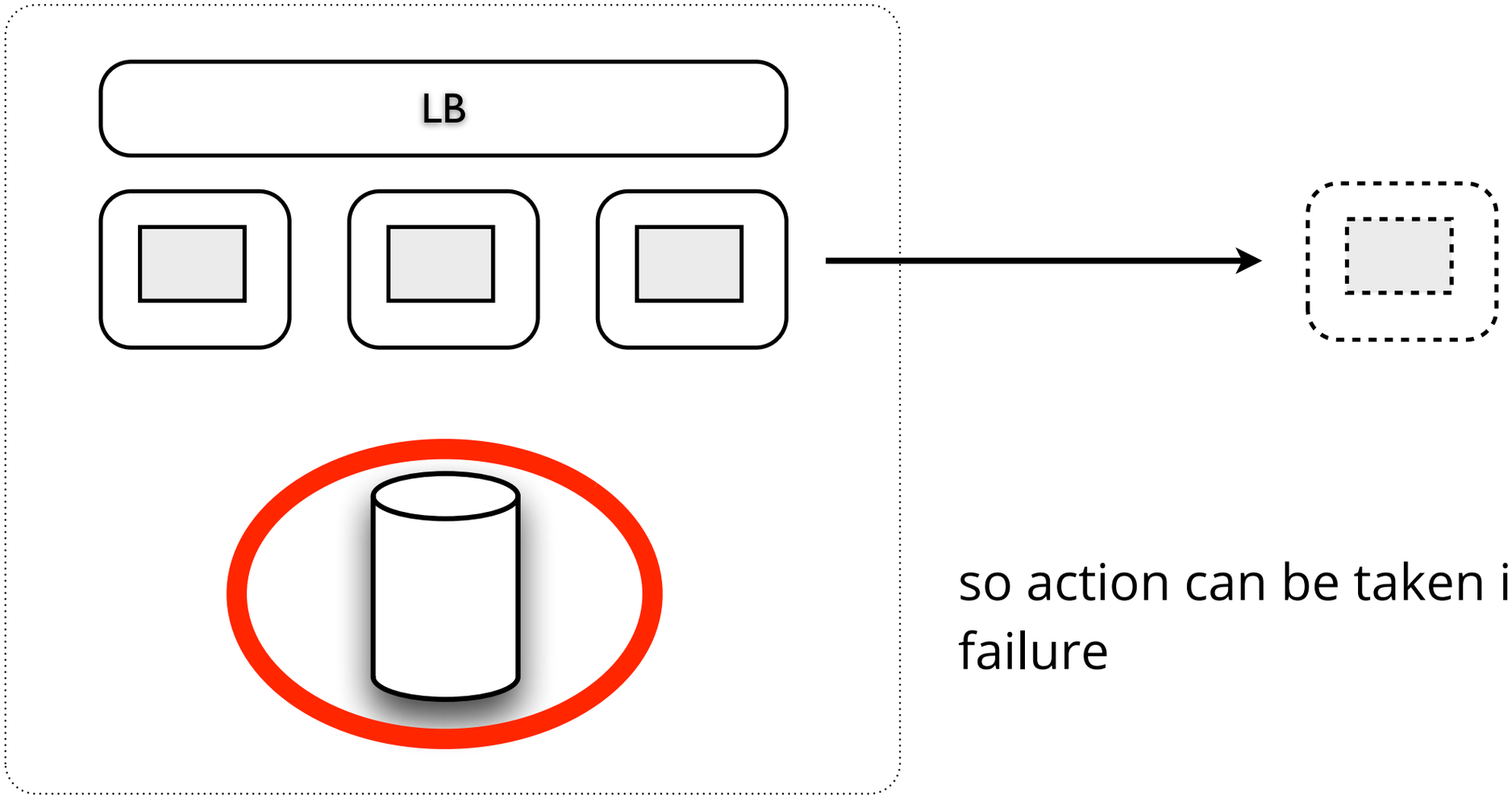


downstream systems and integration points should be monitored by each service



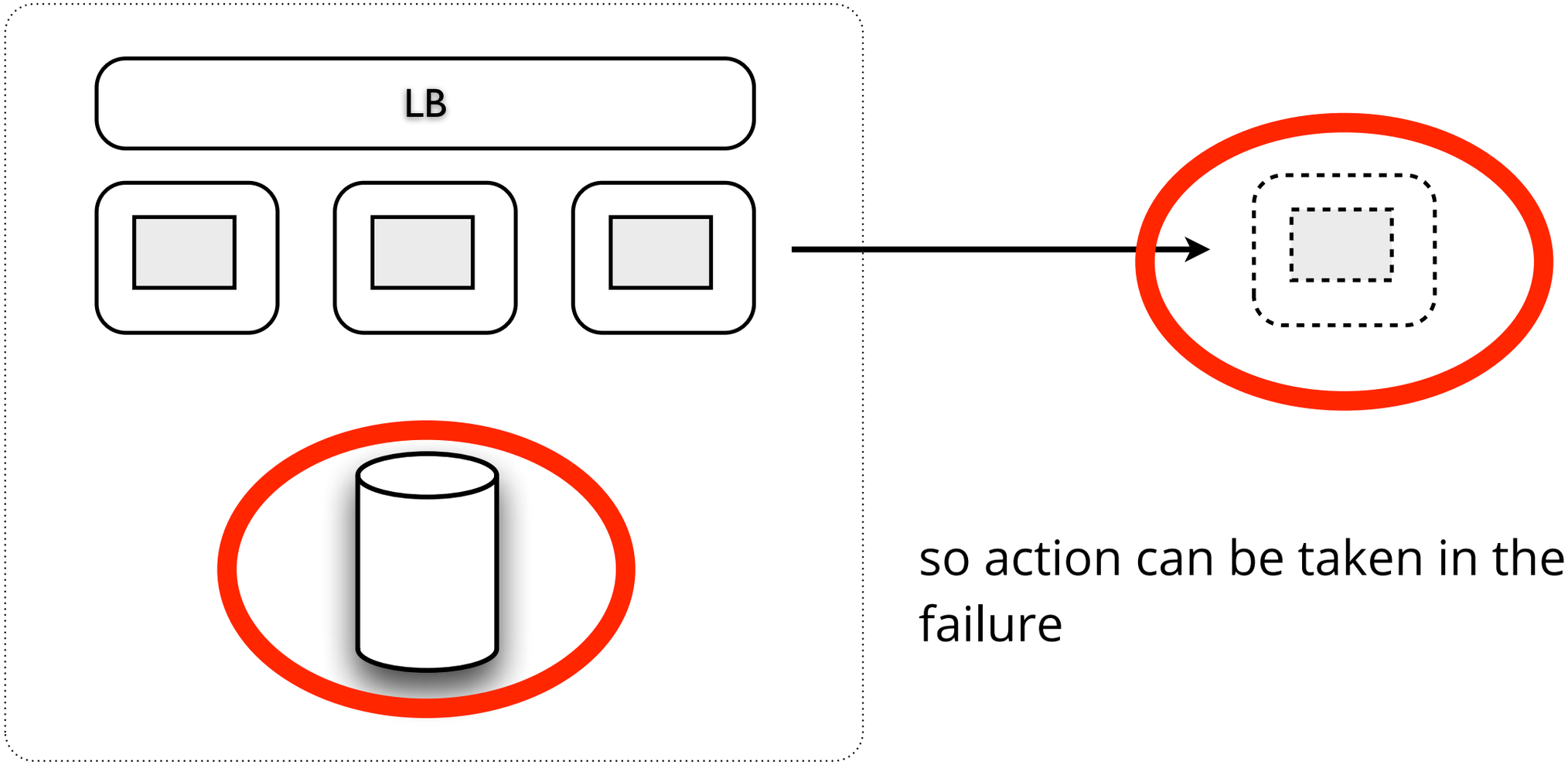
so action can be taken in the result of a failure

downstream systems and integration points should be monitored by each service



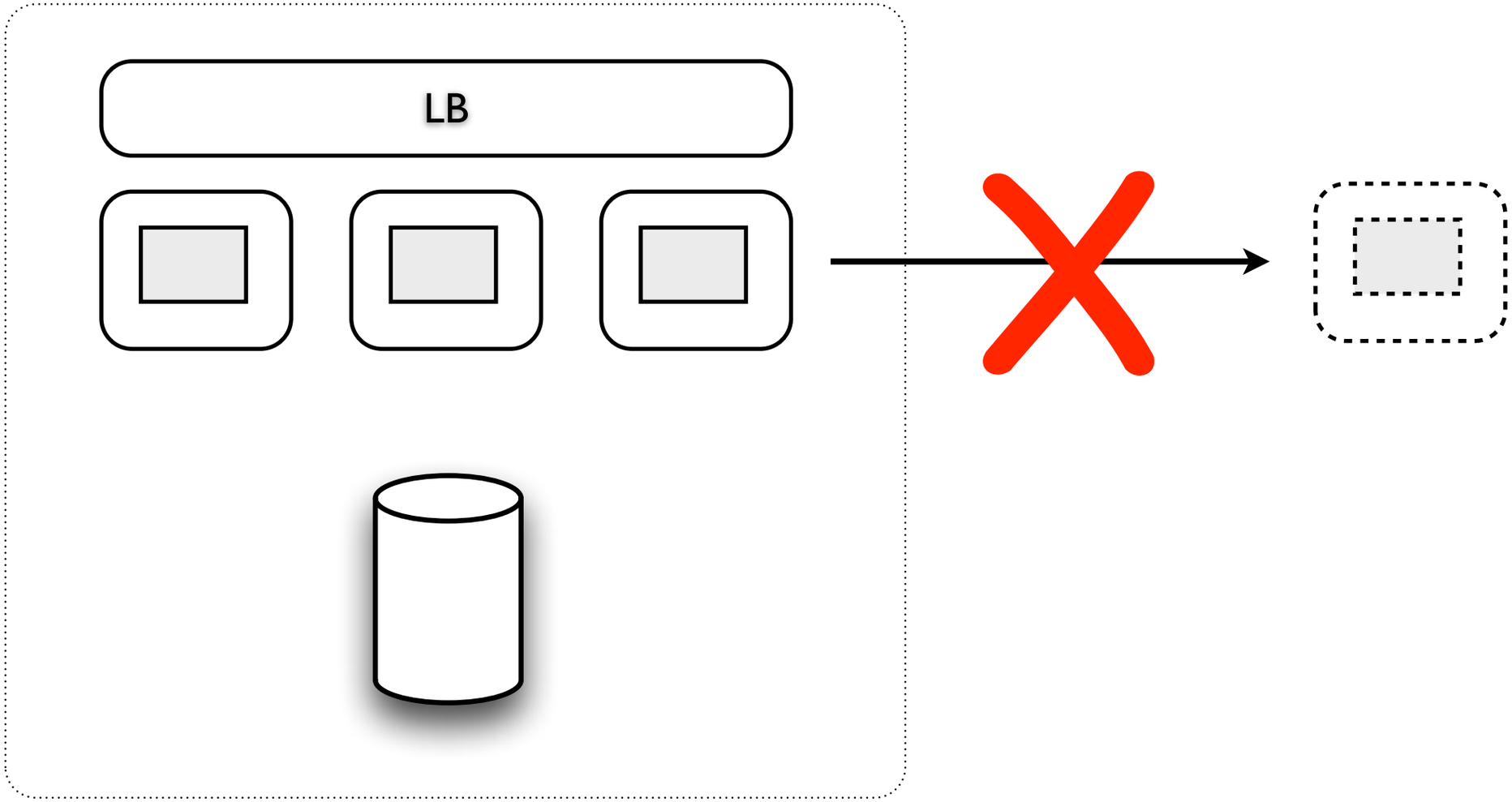
so action can be taken in the result of a failure

downstream systems and integration points should be monitored by each service

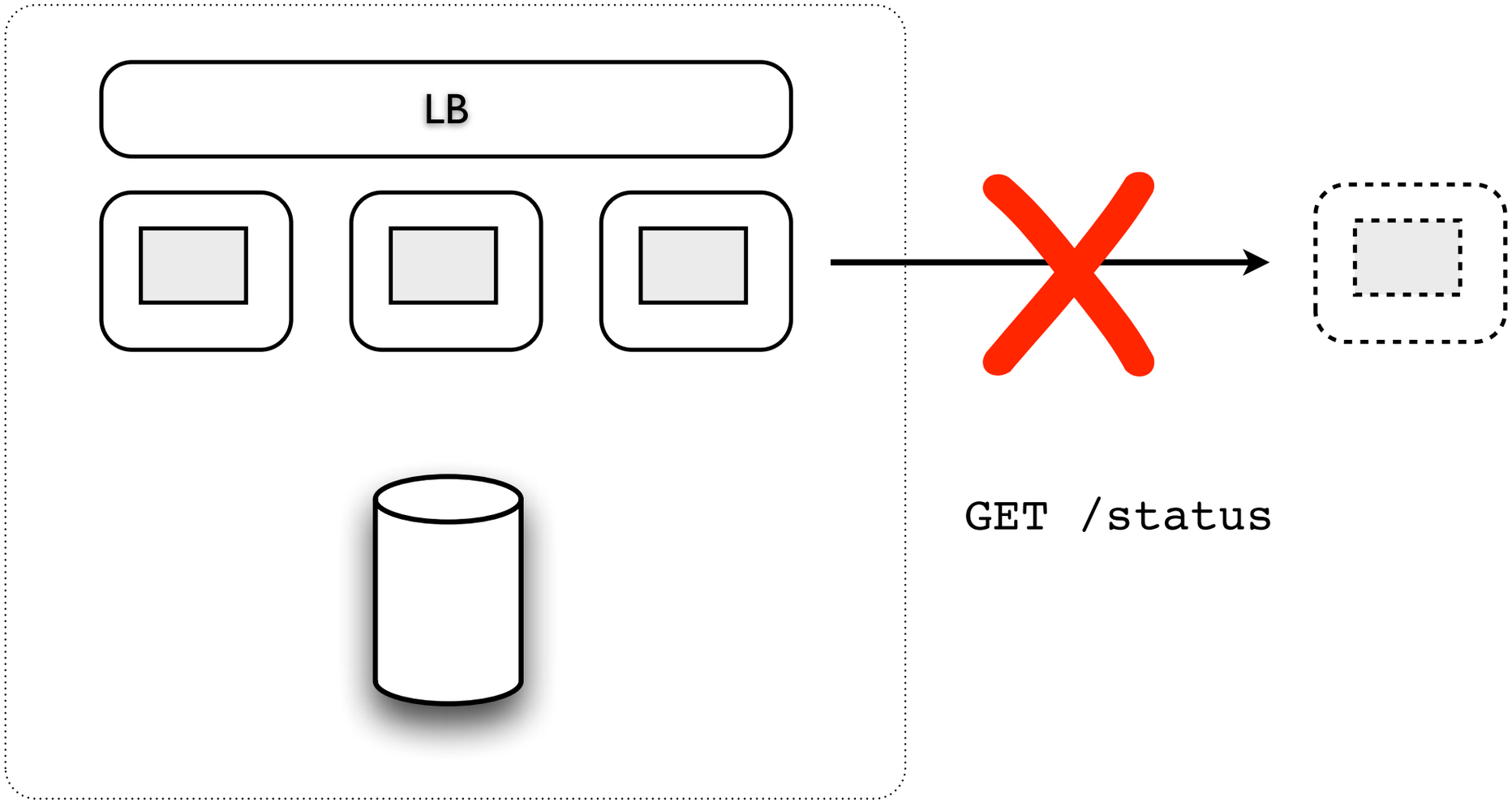


so action can be taken in the result of a failure

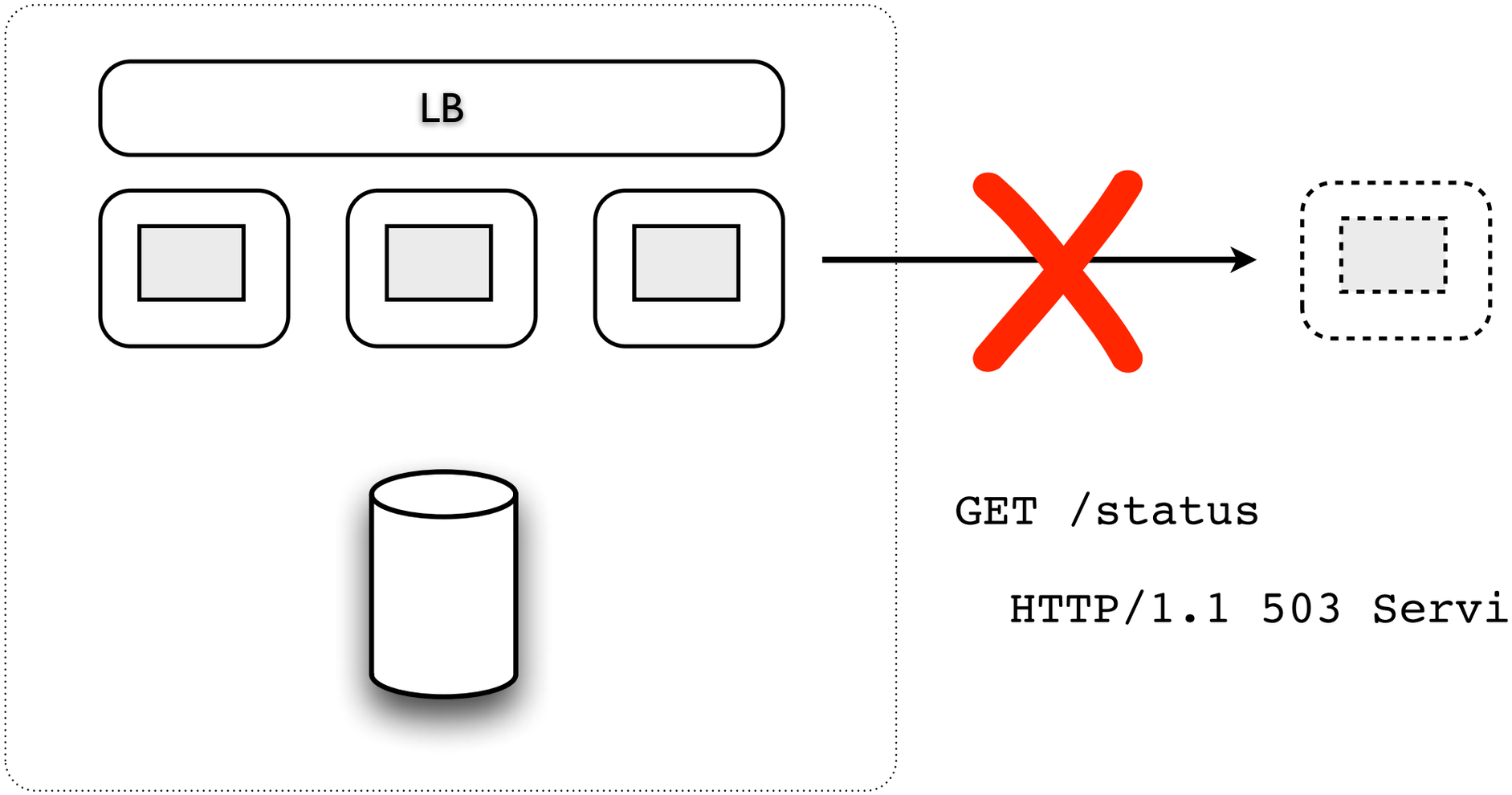
for example, if a downstream service fails



for example, if a downstream service fails

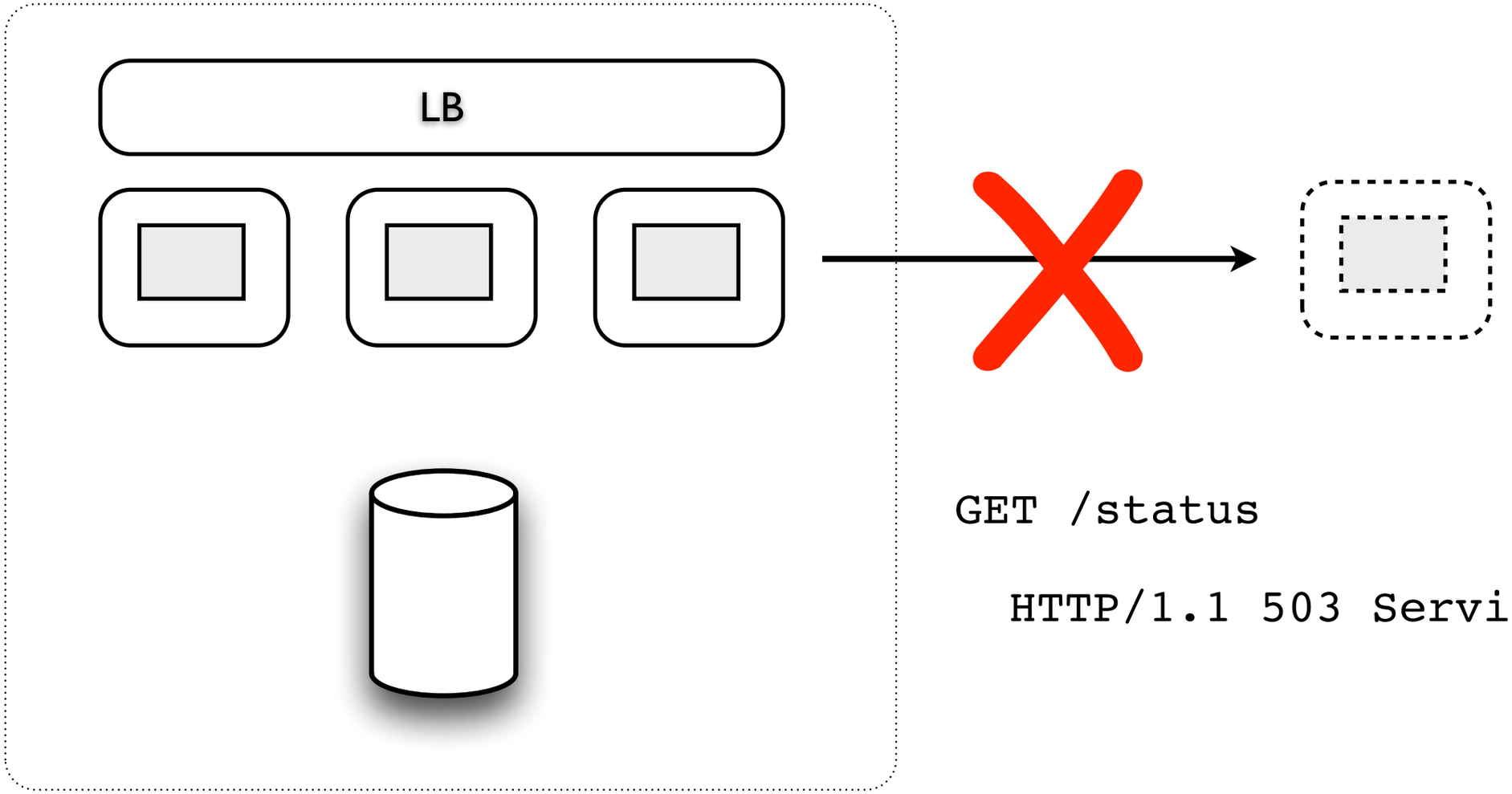


for example, if a downstream service fails




```
GET /status

  HTTP/1.1 503 Service Unavailable
```

for example, if a downstream service fails



the application could block incoming requests, or notify operations

in java land there is a nice library from the good folks at yammer that allows you to do a lot of this

```
import com.yammer.metrics.core.HealthCheck;

public class NamedHealthCheck extends HealthCheck {

    private final Pingable pingable;
    private final String name;

    public NamedHealthCheck(Pingable pingable, String name) {
        super(name);
        this.pingable = pingable;
        this.name = name;
    }

    @Override
    protected Result check() {
        try {
            return (pingable.ping())
                    ? Result.healthy()
                    : Result.unhealthy(String.format("Could not connect to the %s", name));
        } catch (Exception e) {
            return Result.unhealthy(String.format("Could not connect to the %s", name));
        }

    }

}
```

http://metrics.codahale.com

then there are the groovy new (and some not so new) tools on the block

then there are the groovy new (and some not so new) tools on the block

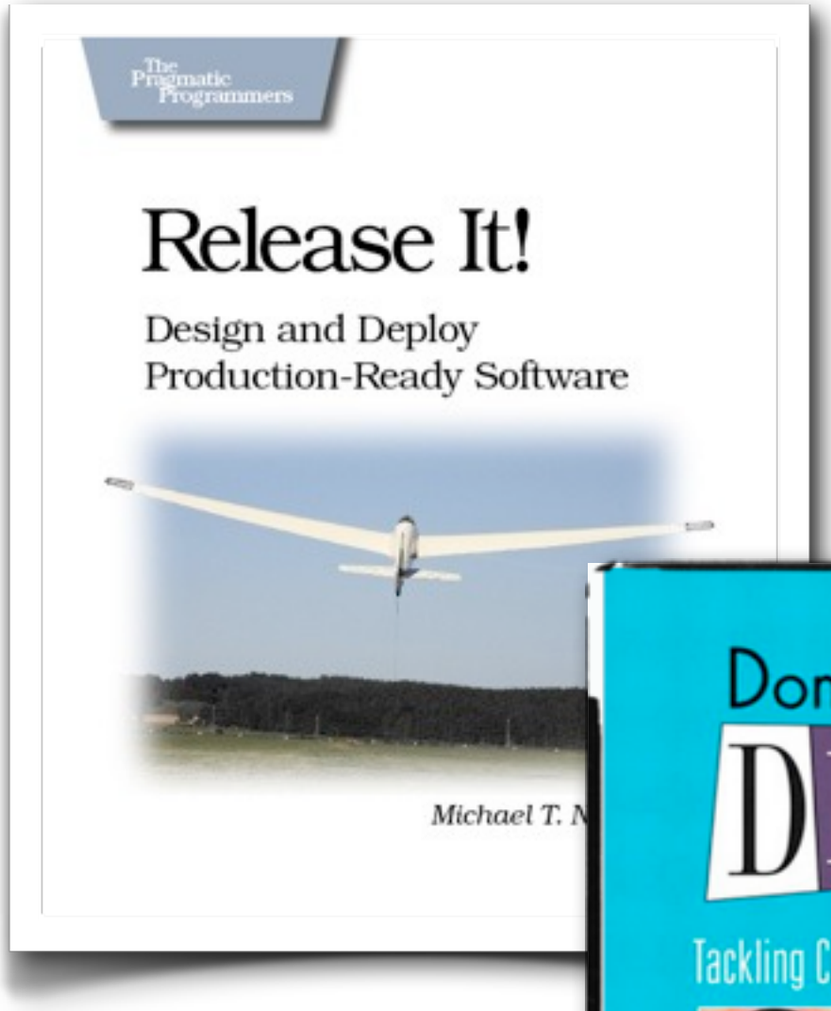
- Graylog2
- Syslog-ng
- Logstash
- Scribe
- Splunk
- riemann

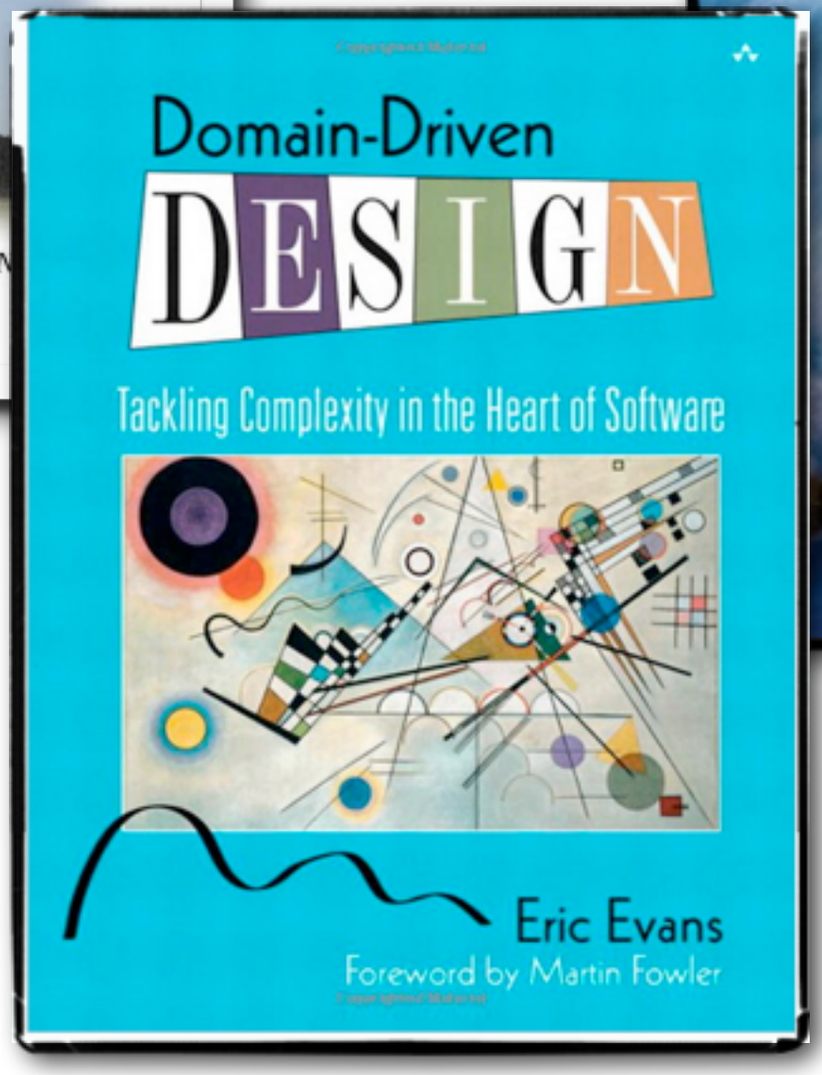
then there are the groovy new (and some not so new) tools on the block

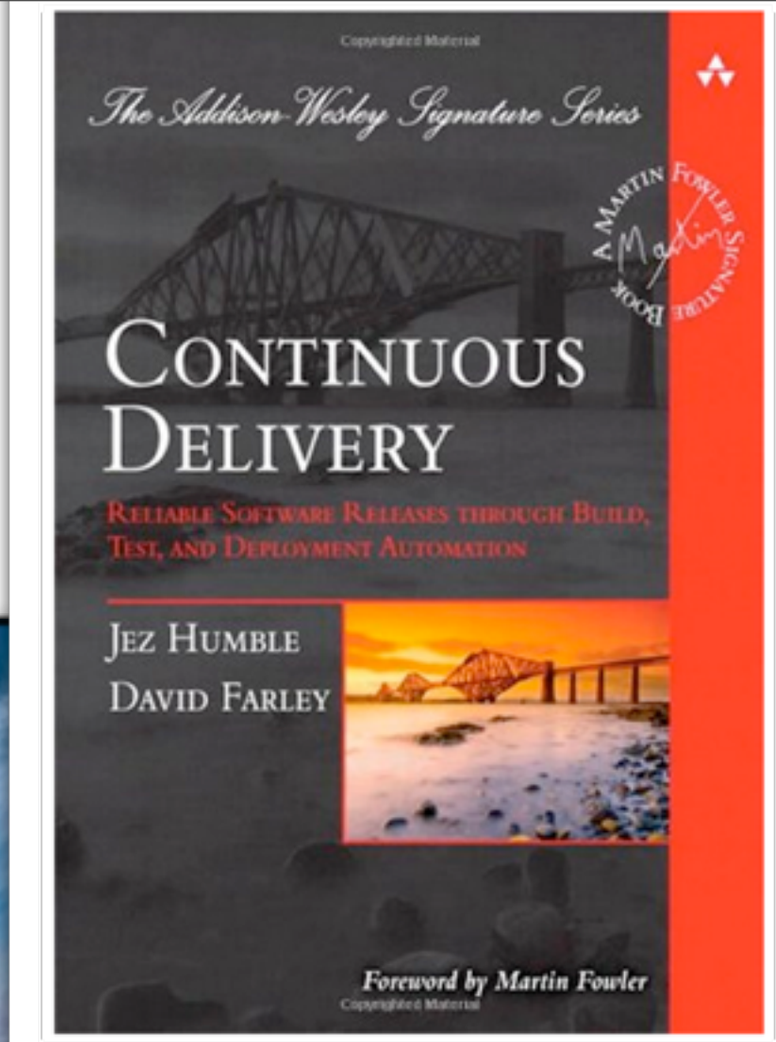
- Graylog2
- Syslog-ng
- Logstash
- Scribe
- Splunk
- riemann

and a final shout out to zipkin for distributed system monitoring

standing on the
shoulders of giants


The Pragmatic Programmers
Release It!
Design and Deploy
Production-Ready Software
Michael T. N


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous
Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler


REST in Practice

but "invented a slightly better one. That finally got changed once more to what we have today. He put pipes into Unix." Thompson also had to change most of the programs, because up until that time, they couldn't take standard input. There wasn't really a need; they all had file arguments. "GREP had a file argument, CAT had a file argument."

The next morning, "we had this orgy of `one liners.' Everybody had a one liner. Look at this, look at that. ...Everybody started putting forth the UNIX philosophy. Write programs that do one thing and do it well. Write programs to work together. Write programs that handle text streams, because that is a universal interface." Those ideas which add up to the tool approach, were there in some unformed way before pipes, but they really came together afterwards. Pipes became the catalyst for this UNIX philosophy. "The tool thing has turned out to be actually successful. With pipes, many programs could work together, and they could work together at a distance."

The Unix Philosophy

Lions commentary on Unix 2nd edition

to build systems is to make trade-offs

to build systems is to make trade-offs

maintainability vs time-to-market

throughput vs cost

portability vs deployability

to build systems is to make trade-offs

maintainability vs time-to-market

throughput vs cost

portability vs deployability

infrastructure automation and tooling is essential to get the benefits of this approach

thanks

jalewis@thoughtworks.com : @boicy : http://bovon.org

BERLIN
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2013

goto; conference

Training:Oct 15-16 / Conference:Oct 17-18

ThoughtWorks®

is hiring

jalewis@thoughtworks.com : @boicy : http://bovon.org