
APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 25-27 SEPTEMBER 2017

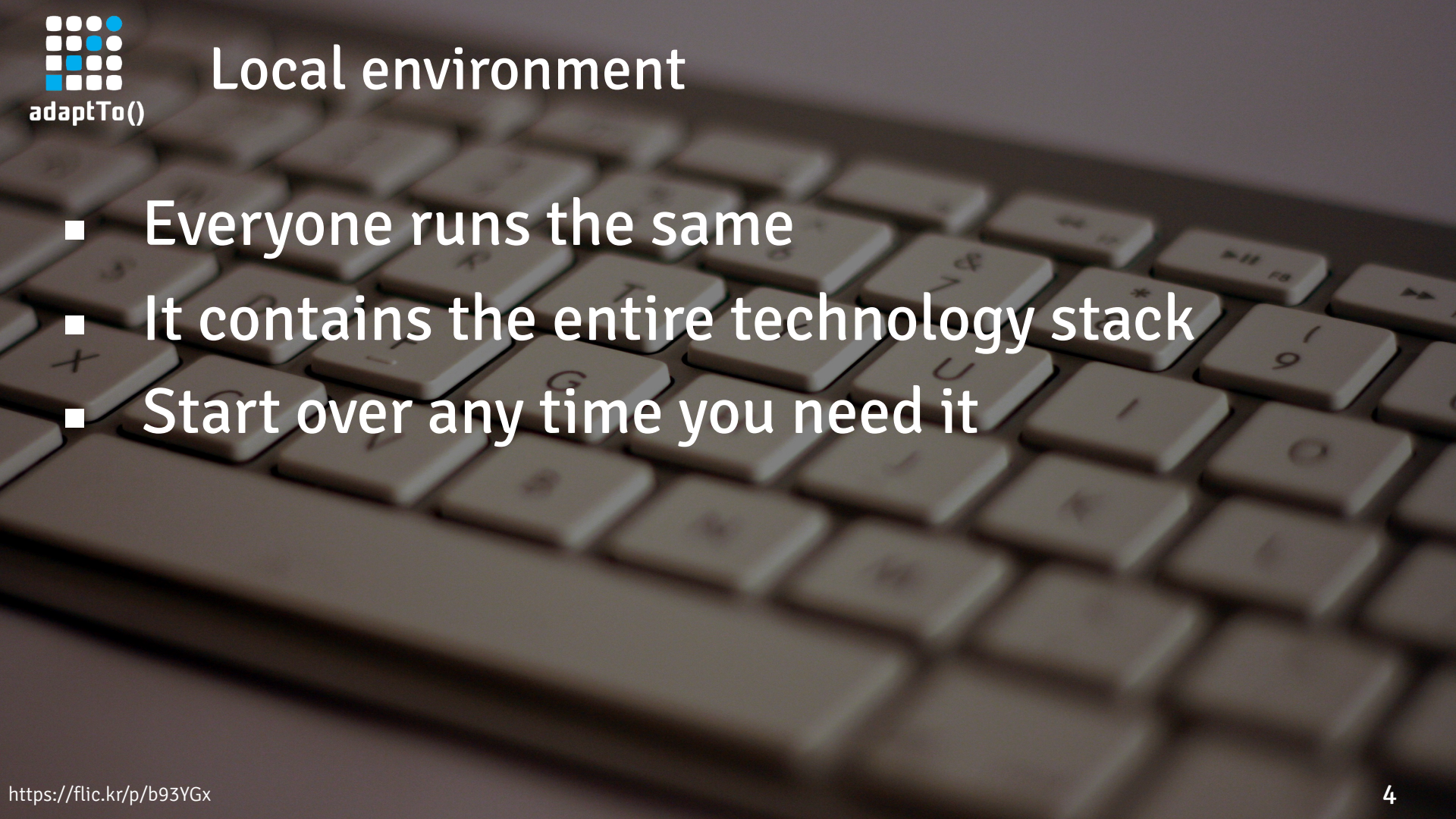
# Taming AEM deployments

Jakub Wądołowski, Cognifide


adaptTo()
The road to efficient AEM delivery
Starts on your computer
Ends in production
Includes everything in between
https://flic.kr/p/z8g25R
2

adaptTo()

# Development phase



# Local environment

- Everyone runs the same
- It contains the entire technology stack
- Start over any time you need it

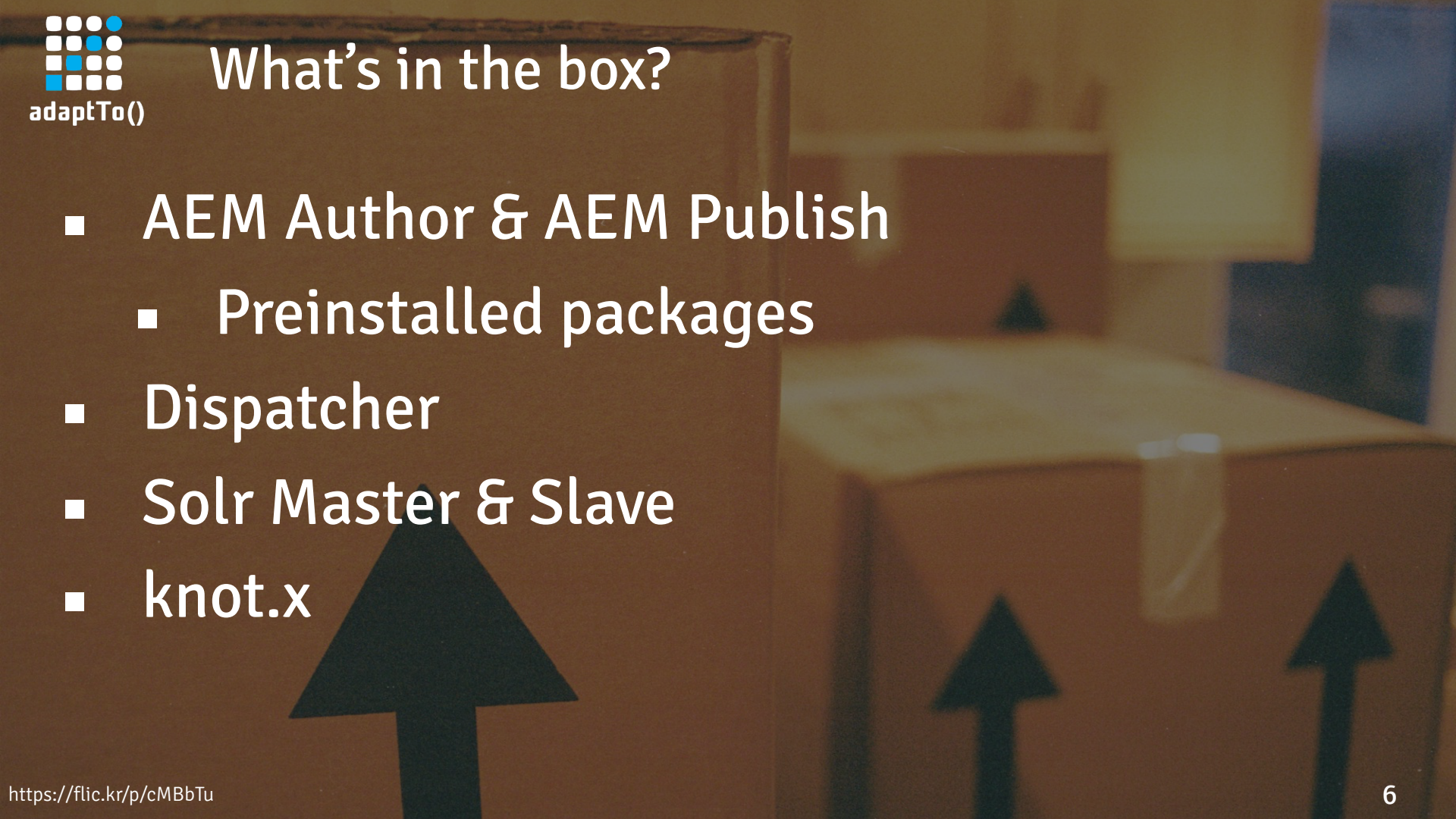
https://flic.kr/p/b93YGx

adaptTo()

# Solution

HashiCorp

Vagrant


adaptTo()
What's in the box?
AEM Author & AEM Publish
Preinstalled packages
Dispatcher
Solr Master & Slave
knot.x
https://flic.kr/p/cMBbTu
6


# Vagrant in practice (1/3)

```
# Install prerequisites:
# * Vagrant
# * VirtualBox
# * ChefDK
$ git clone git@example.org:xyz/vagrant.git
$ cd project-vagrant
$ rake init
$ rake up
```


# Vagrant in practice (2/3)

- http://localhost:4502 (AEM Author)
- http://localhost:4503 (AEM Publish)
- https://example.vagrant (Dispatcher)
- http://localhost:8983 (Solr)
- etc


# Vagrant in practice (3/3)

```
$ rake provision:publish
...
Chef Client finished, 8/196 resources updated in 02
minutes 02 seconds

$ rake provision:httpd
...
Chef Client finished, 4/41 resources updated in 19
seconds
```


# Vagrant talk

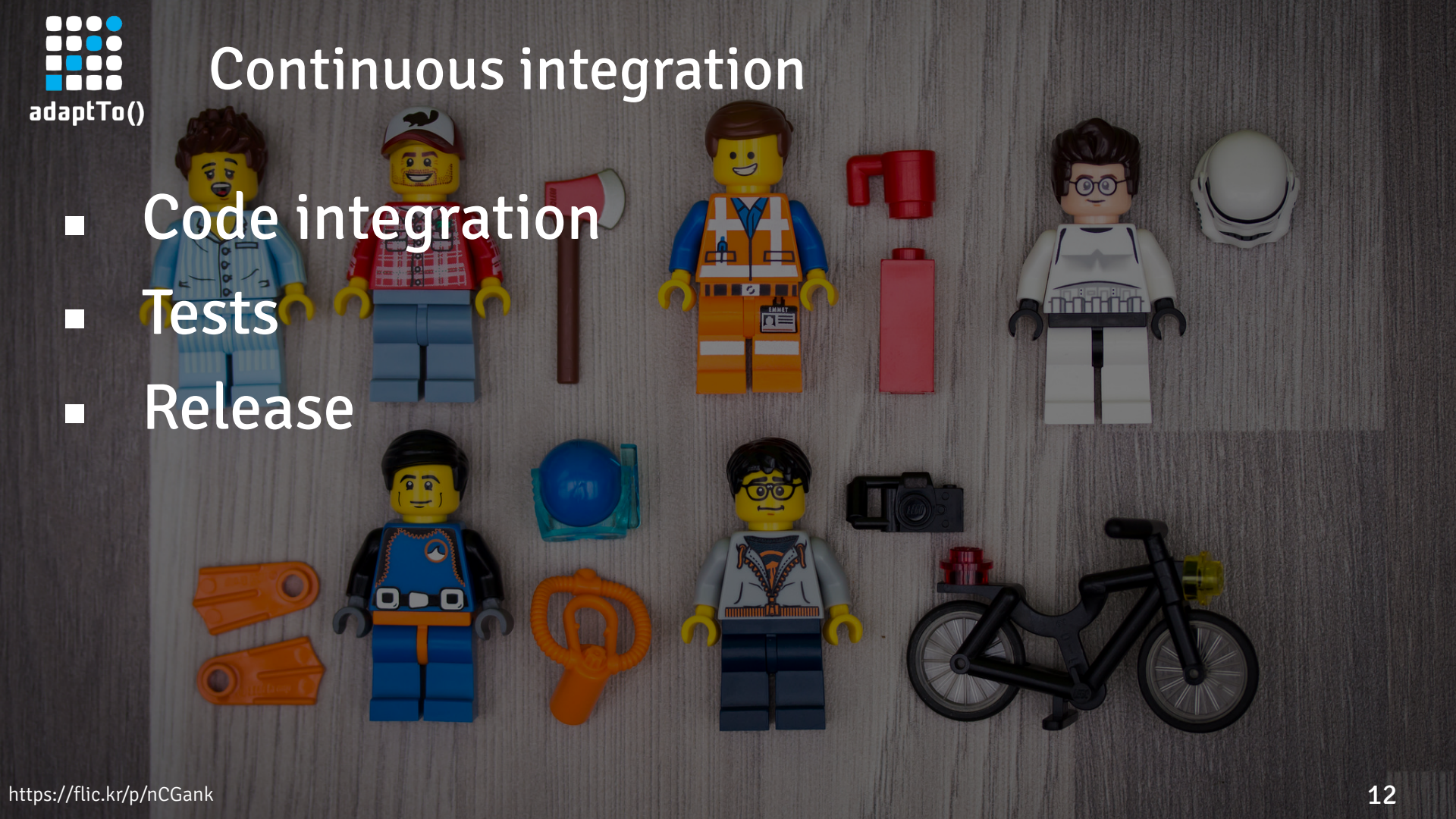
https://goo.gl/hu216e


# Continuous integration

adaptTo()

# Continuous integration

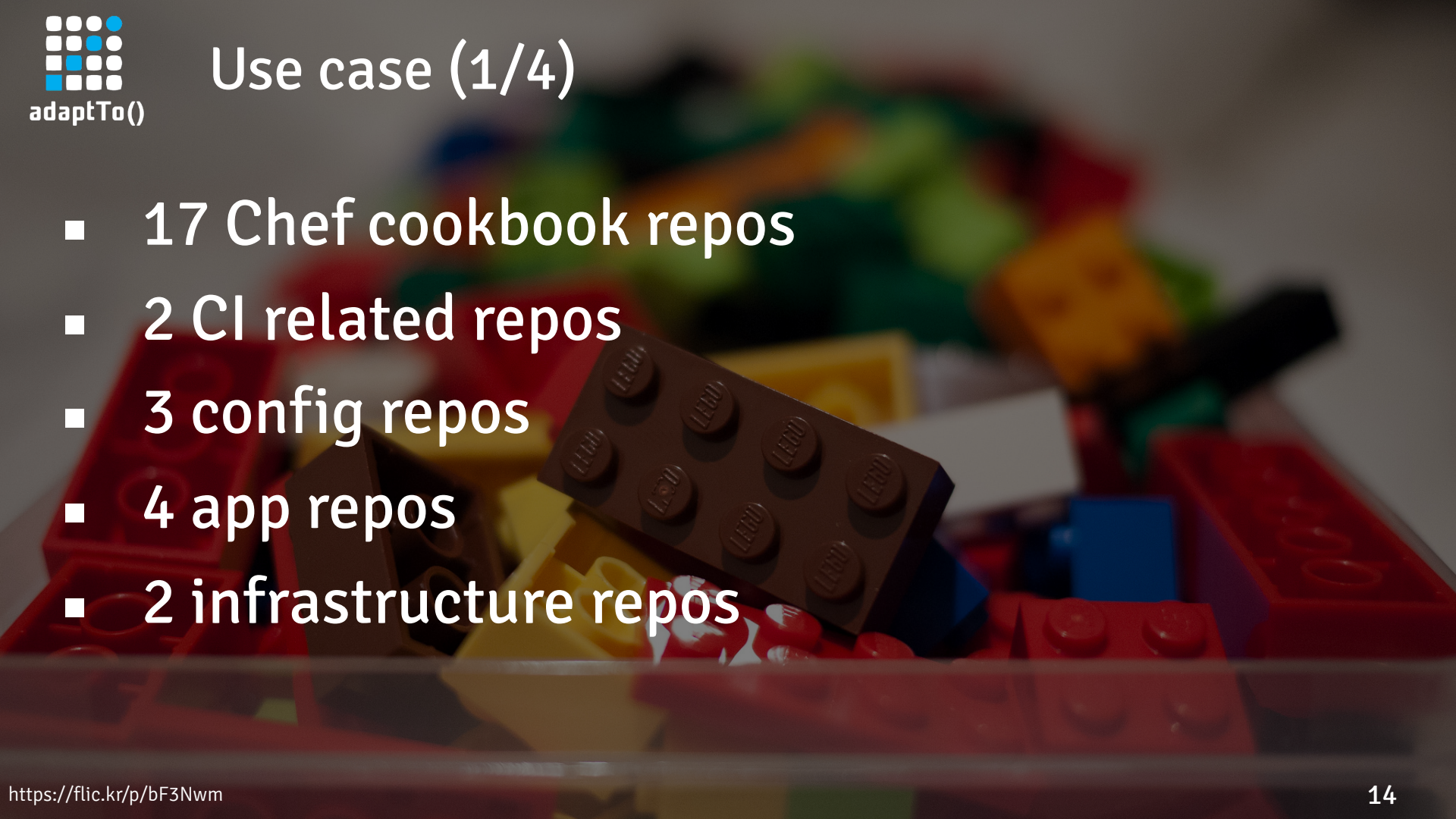
- Code integration
- Tests
- Release



https://flic.kr/p/nCGank

adaptTo()

# Git flow



# Use case (1/4)

- 17 Chef cookbook repos
- 2 CI related repos
- 3 config repos
- 4 app repos
- 2 infrastructure repos

https://flic.kr/p/bF3Nwm

adaptTo()

# Use case (2/4)

- Each repository has its own lifecycle
- Simple Git flow

https://flic.kr/p/peZfvA

# Use case (3/4)

```
$ tree -L 1
.
├── aem
```

https://flic.kr/p/fnPQuz

# Use case (3/4)

```
$ tree -L 1
.
├── aem
├── domain
```

https://flic.kr/p/fnPQuz

# Use case (3/4)

```
$ tree -L 1
.
├── aem
├── domain
├── knotx-autocomplete
├── knotx-download
├── knotx-filters
├── knotx-index
└── knotx-search
```

https://flic.kr/p/fnPQuz

# Use case (3/4)

```
$ tree -L 1
.
├── aem
├── domain
├── knotx-autocomplete
├── knotx-download
├── knotx-filters
├── knotx-index
├── knotx-search
├── solr
```

https://flic.kr/p/fnPQuz

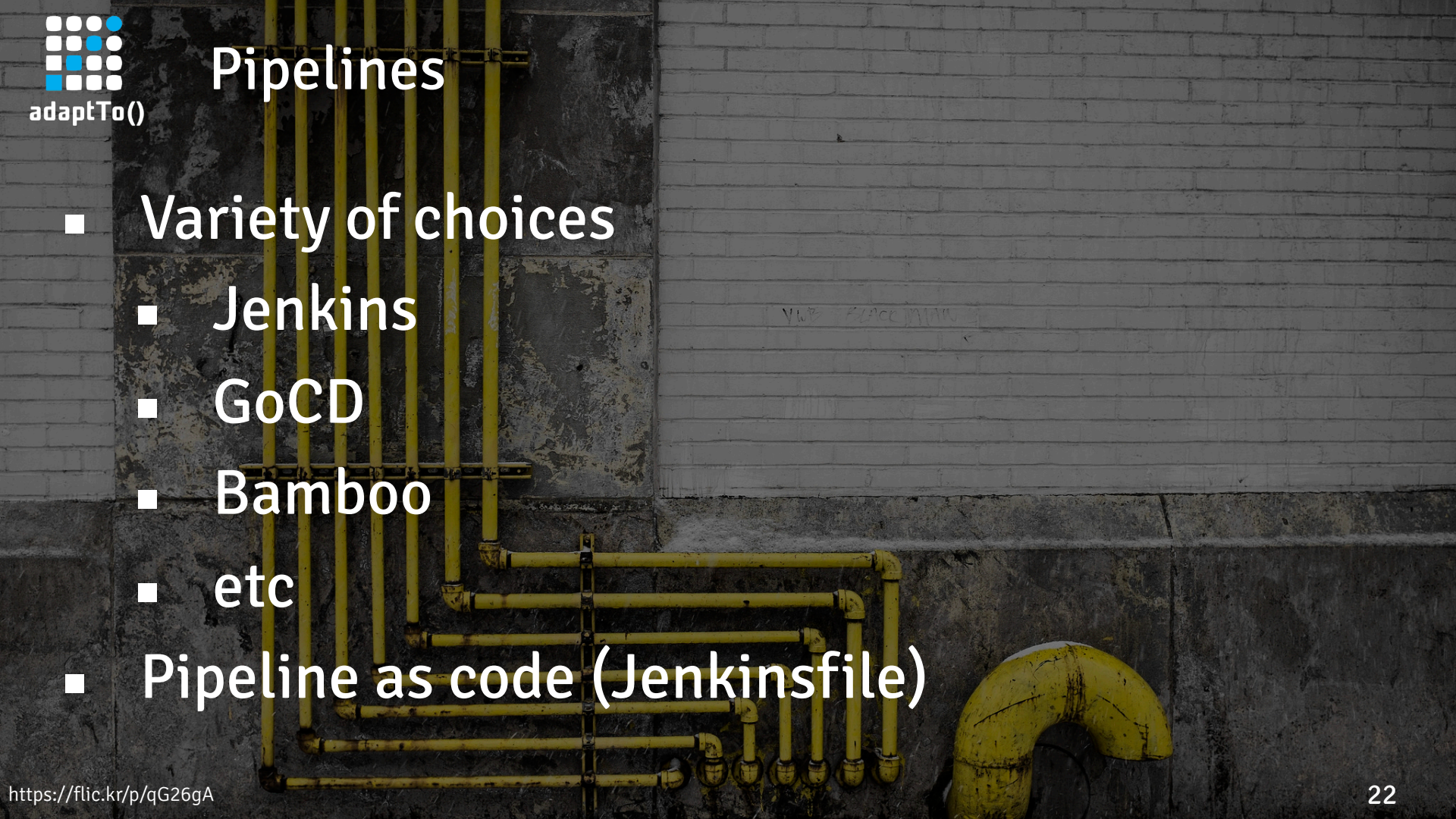
# Use case (3/4)

```
$ tree -L 1
.
├── aem
├── domain
├── knotx-autocomplete
├── knotx-download
├── knotx-filters
├── knotx-index
├── knotx-search
├── solr
└── pom.xml
```

https://flic.kr/p/fnPQuz


# Use case (4/4)

- Update of 1 element impacts all the remaining ones
- End-to-end testing
- 1 version & multiple artifacts



# Pipelines

- Variety of choices
  - Jenkins
  - GoCD
  - Bamboo
  - etc
- Pipeline as code (Jenkinsfile)

https://flic.kr/p/qG26gA

```
node('master') {
    stage('Checkout') {









    }
    stage('Build') {









    }
    stage('Test') {



    }
}
```

```
node('master') {
    stage('Checkout') {
        dir('app') {
            git url: 'git://example.com/aem-project.git', branch: 'master'
        }

        dir('tests') {
            git url: 'git://example.com/tests.git', branch: 'master'
        }
    }

    stage('Build') {









    }

    stage('Test') {



    }
}
```

```
node('master') {
    stage('Checkout') {
        dir('app') {
            git url: 'git://example.com/aem-project.git', branch: 'master'
        }

        dir('tests') {
            git url: 'git://example.com/tests.git', branch: 'master'
        }
    }

    stage('Build') {
        node('aem') {
            dir('app/aem') {
                sh 'mvn clean install'
            }
        }
    }

    stage('Test') {



    }
}
```

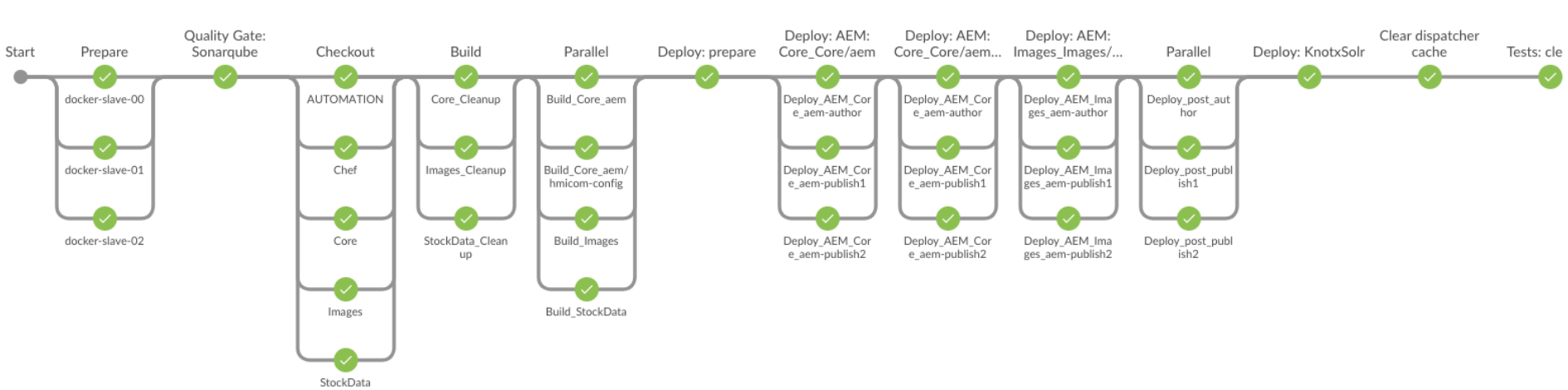
```
node('master') {
    stage('Checkout') {
        dir('app') {
            git url: 'git://example.com/aem-project.git', branch: 'master'
        }

        dir('tests') {
            git url: 'git://example.com/tests.git', branch: 'master'
        }
    }

    stage('Build') {
        node('aem') {
            dir('app/aem') {
                sh 'mvn clean install'
            }
        }
    }

    stage('Test') {
        dir('tests') {
            sh './gradlew clean contractTests'
        }
    }
}
```

# Build / Deploy / Test




## Input required

Deploy all applications (overrides other options and builds from default branches)

- [ ] All

Deploy Core repository

- [ ] Core

Branch in Core repository

develop

Deploy Images application

- [ ] Images

Branch in Images repository

develop

Deploy Models application

- [ ] Models

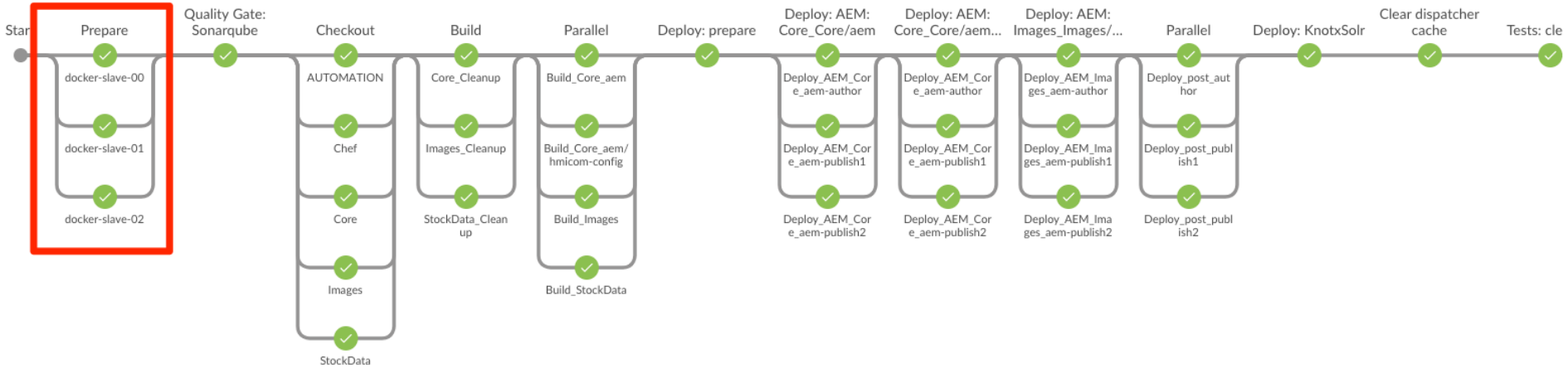
Branch in Models repository

develop

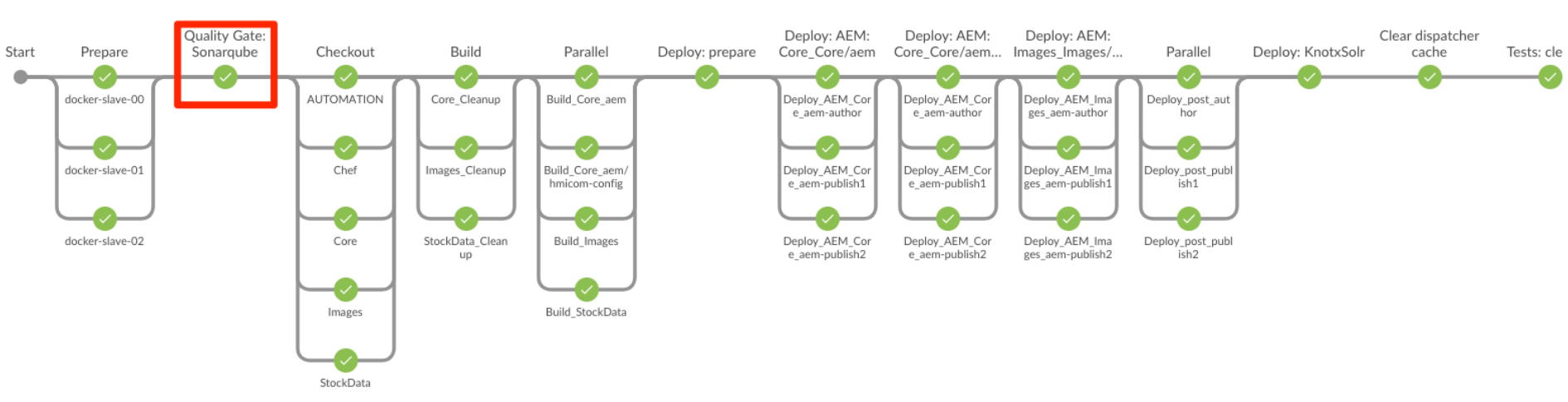
Deploy Stock Data application

Run Cancel

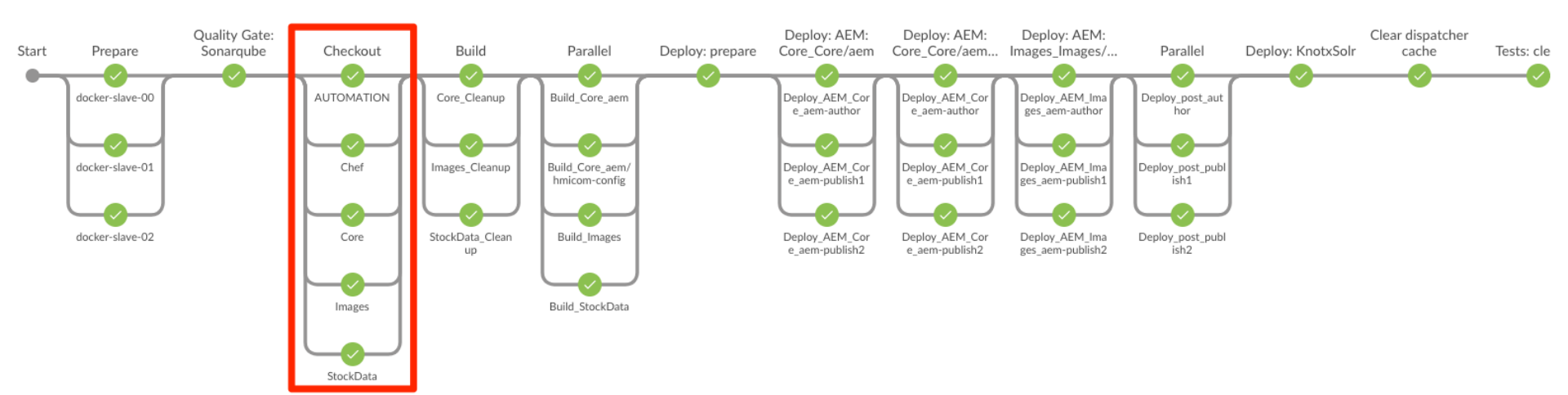
# Build / Deploy / Test



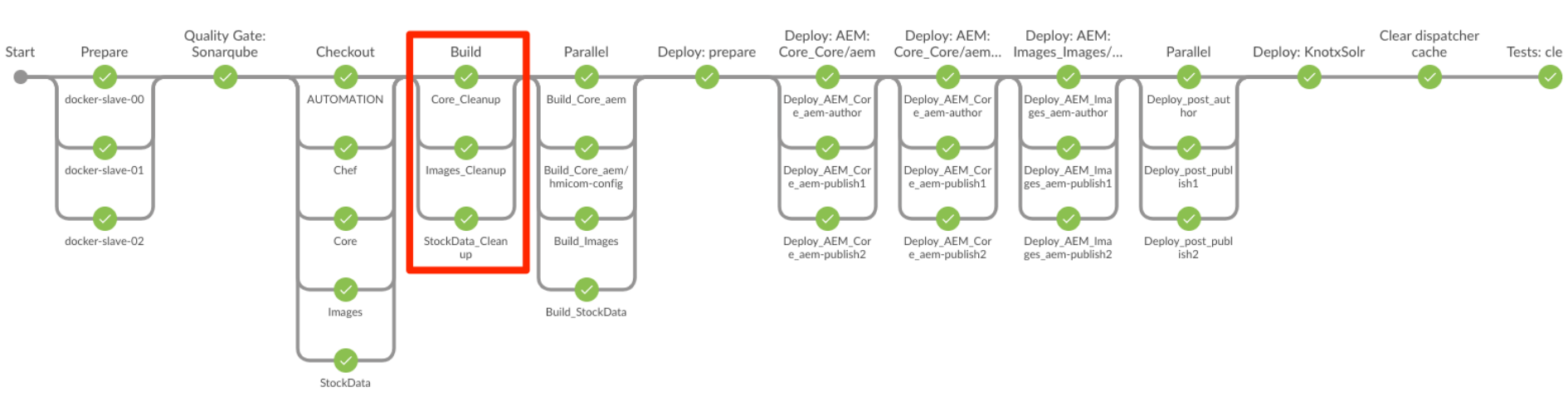
# Build / Deploy / Test



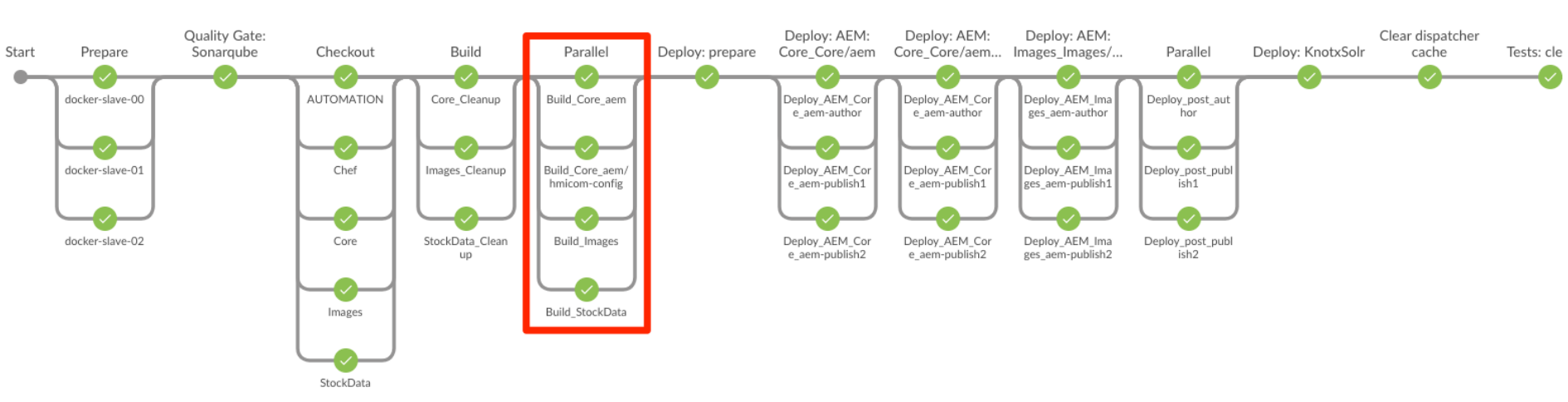
# Build / Deploy / Test



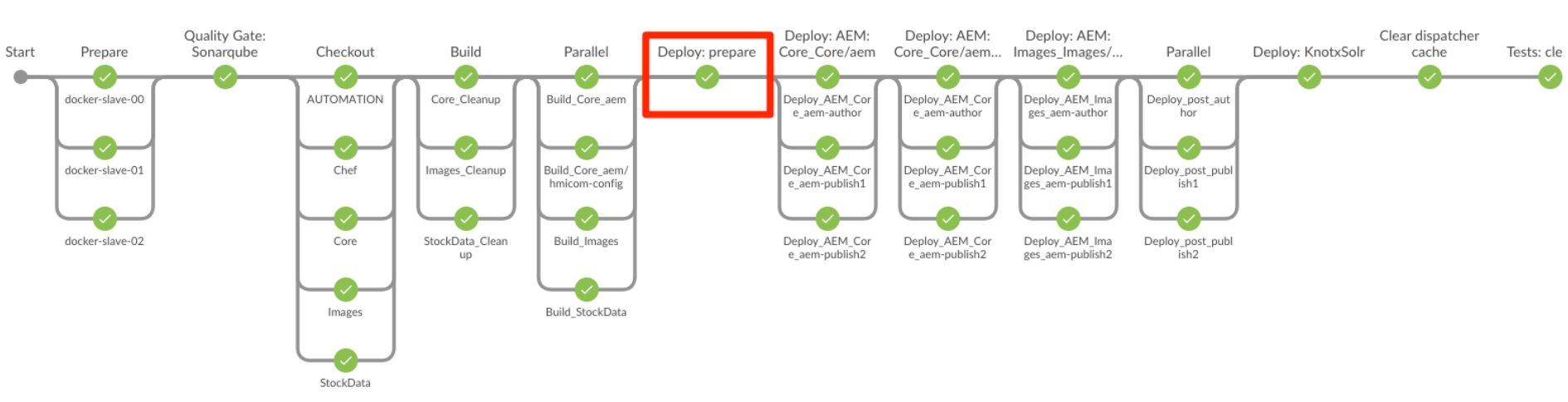
# Build / Deploy / Test



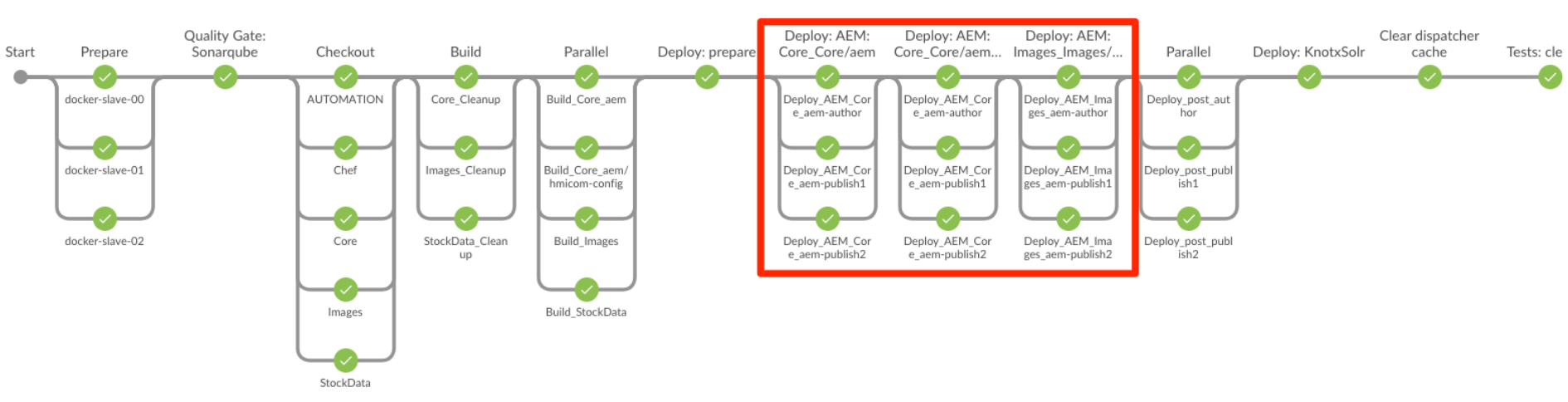
# Build / Deploy / Test



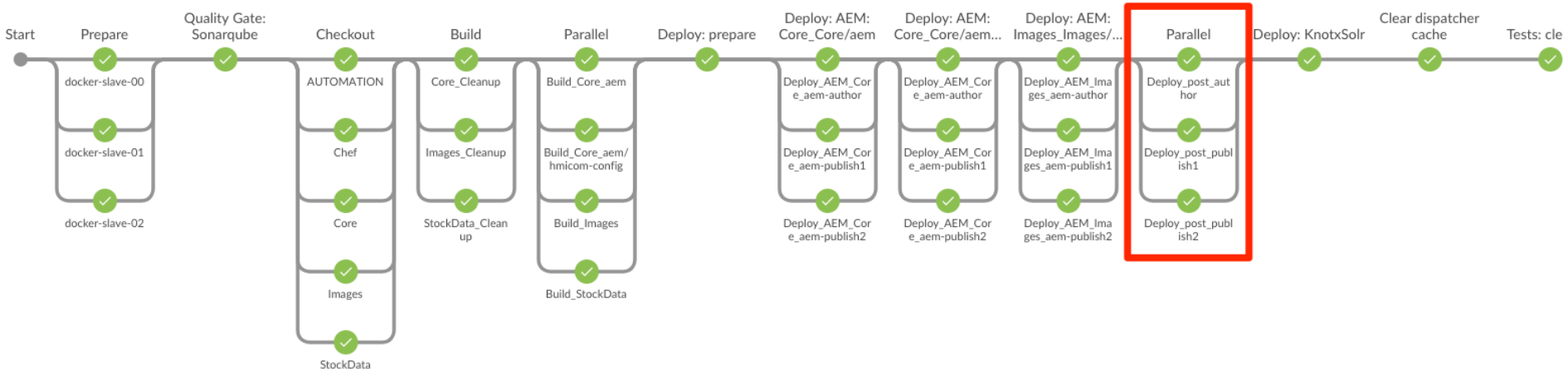
# Build / Deploy / Test



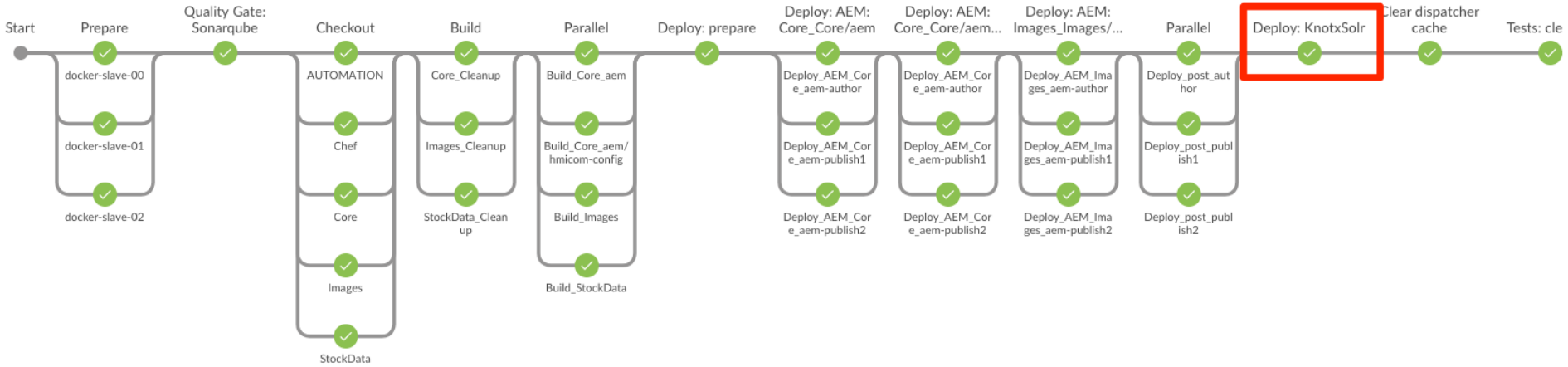
# Build / Deploy / Test



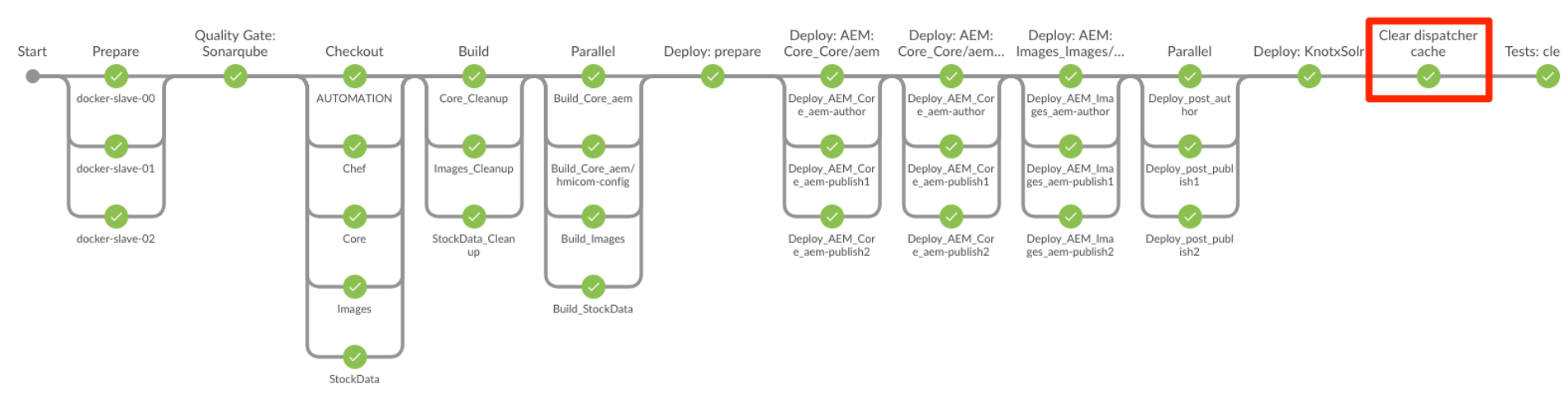
# Build / Deploy / Test



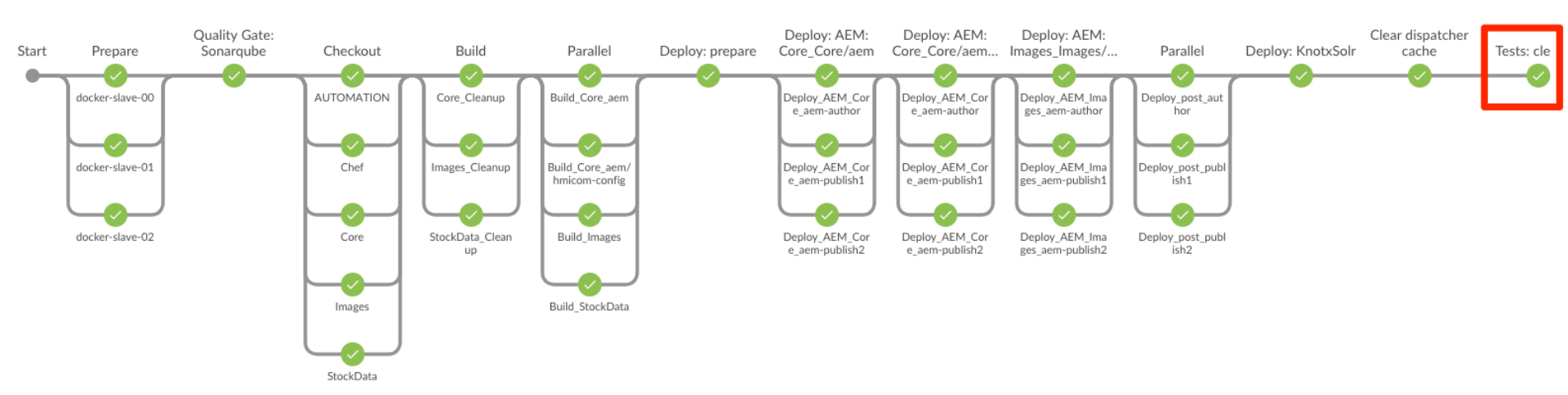
# Build / Deploy / Test



# Build / Deploy / Test

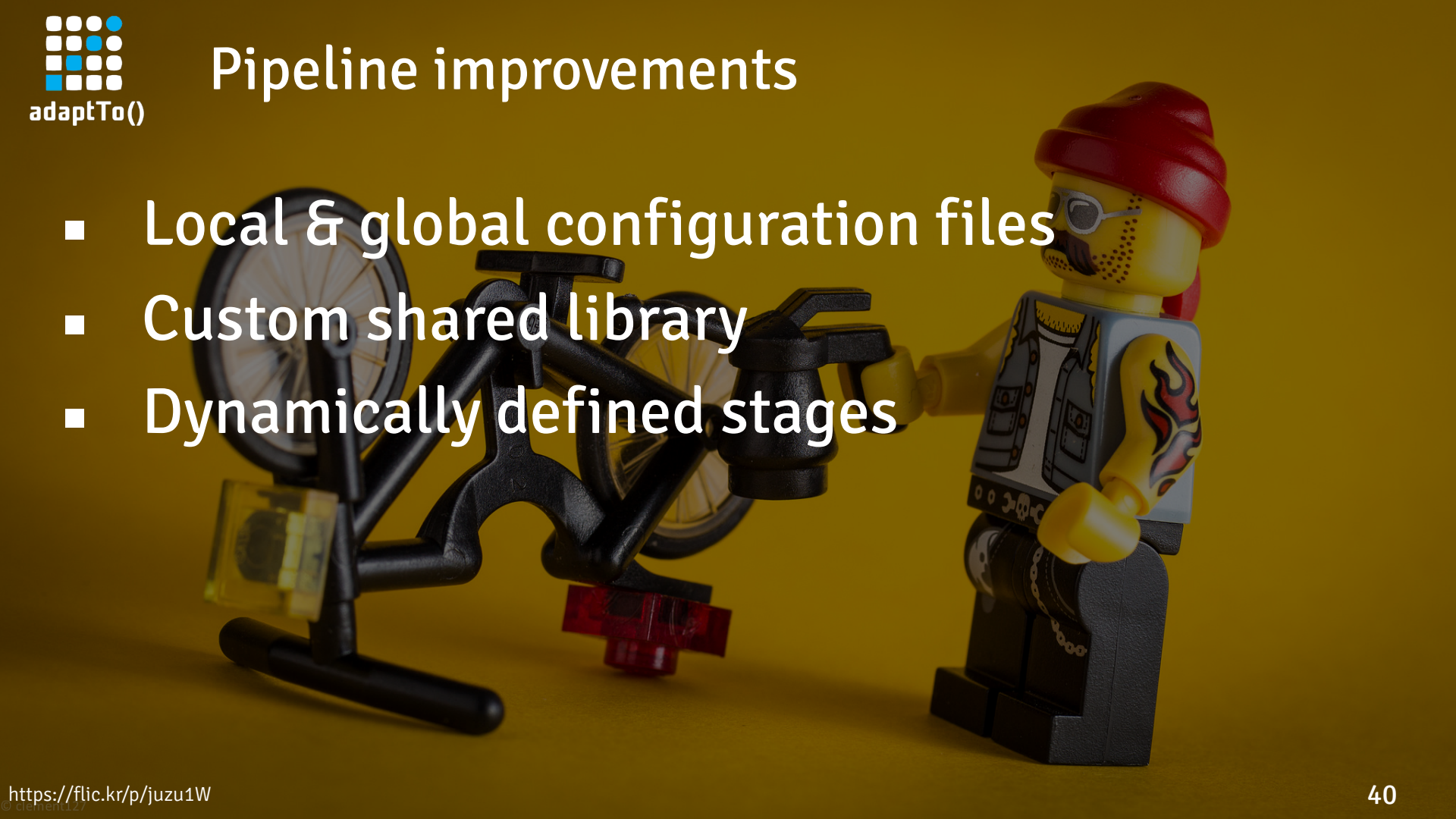


# Build / Deploy / Test



adaptTo()

# Pipeline improvements



- Local & global configuration files
- Custom shared library
- Dynamically defined stages

https://flic.kr/p/juzu1W

```
node('master') {









}
```

```groovy
#!groovy
@Library('cognifide-library@master')
import com.cognifide.jenkinslib.Pipeline

def pipeline = new Pipeline(this)

node('master') {









}
```

```groovy
#!groovy
@Library('cognifide-library@master')
import com.cognifide.jenkinslib.Pipeline

def pipeline = new Pipeline(this)

node('master') {
    withMaven(maven: 'Maven 3') {









    }
}
```

```groovy
#!groovy
@Library('cognifide-library@master')
import com.cognifide.jenkinslib.Pipeline

def pipeline = new Pipeline(this)

node('master') {
    withMaven(maven: 'Maven 3') {
        stage('Quality Gate: Sonarqube') {

        }

        stage('Build') {

        }

        stage('Clear dispatcher cache') {


        }
    }
}
```

```groovy
#!groovy
@Library('cognifide-library@master')
import com.cognifide.jenkinslib.Pipeline

def pipeline = new Pipeline(this)

node('master') {
    withMaven(maven: 'Maven 3') {
        stage('Quality Gate: Sonarqube') {
            sonarqube pipeline.CFG.sonarqube
        }

        stage('Build') {

        }

        stage('Clear dispatcher cache') {


        }
    }
}
```

```groovy
#!groovy
@Library('cognifide-library@master')
import com.cognifide.jenkinslib.Pipeline

def pipeline = new Pipeline(this)

node('master') {
    withMaven(maven: 'Maven 3') {
        stage('Quality Gate: Sonarqube') {
            sonarqube pipeline.CFG.sonarqube
        }

        stage('Build') {
            // ...
        }

        stage('Clear dispatcher cache') {
            knifeExec query: "chef_environment:${pipeline.CFG.chef.environmentName}
AND tags:dispatcher"
        }
    }
}
```

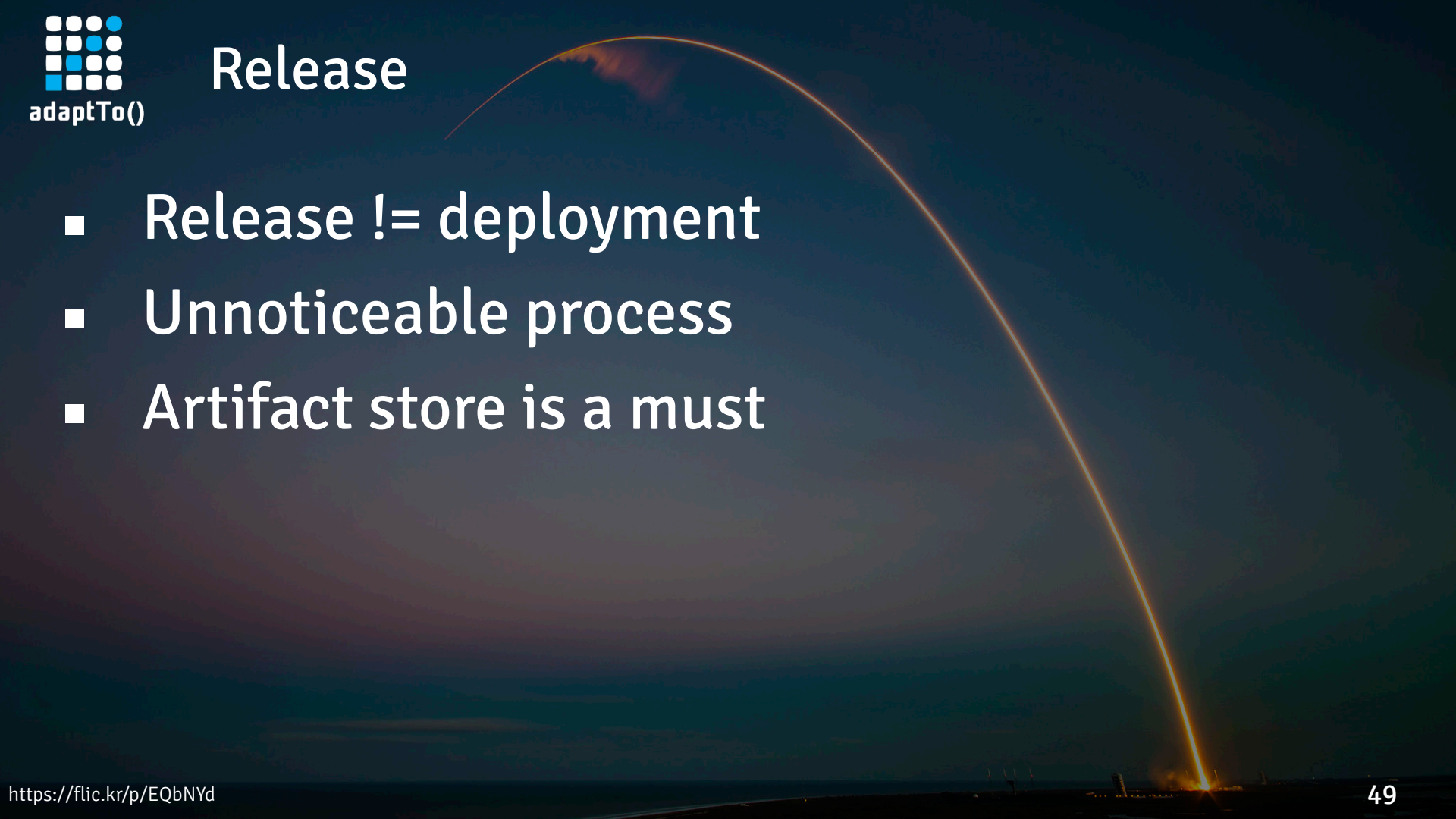
```yaml
notifications:
  enabled: true
  recipients:
    - 'dev-team@example.com'
deploy:
  profiles:
    aem-author: 'dev-author'
    aem-publish1: 'dev-publish1'
    aem-publish2: 'dev-publish2'
tests:
  bobcat:
    profiles:
      - 'dev'
      - 'grid'
    mvnParams: '-Dfork.count=4'
chef:
  environmentName: 'dev'
```

adaptTo()

# Testing

- Unit tests
- Bobcat
  - github.com/Cognifide/bobcat
- Zalenium & Docker
  - github.com/zalando/zalenium
- AET
  - github.com/Cognifide/aet

https://flic.kr/p/juzgT9



# Release

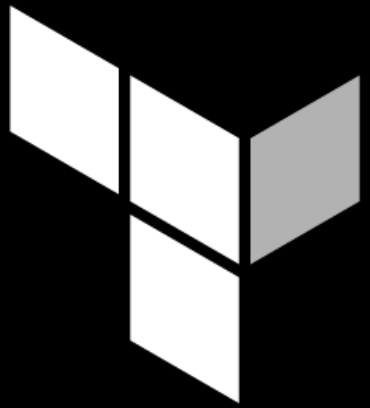
- Release != deployment
- Unnoticeable process
- Artifact store is a must

https://flic.kr/p/EQbNYd


# Infrastructure

adaptTo()

# Solution



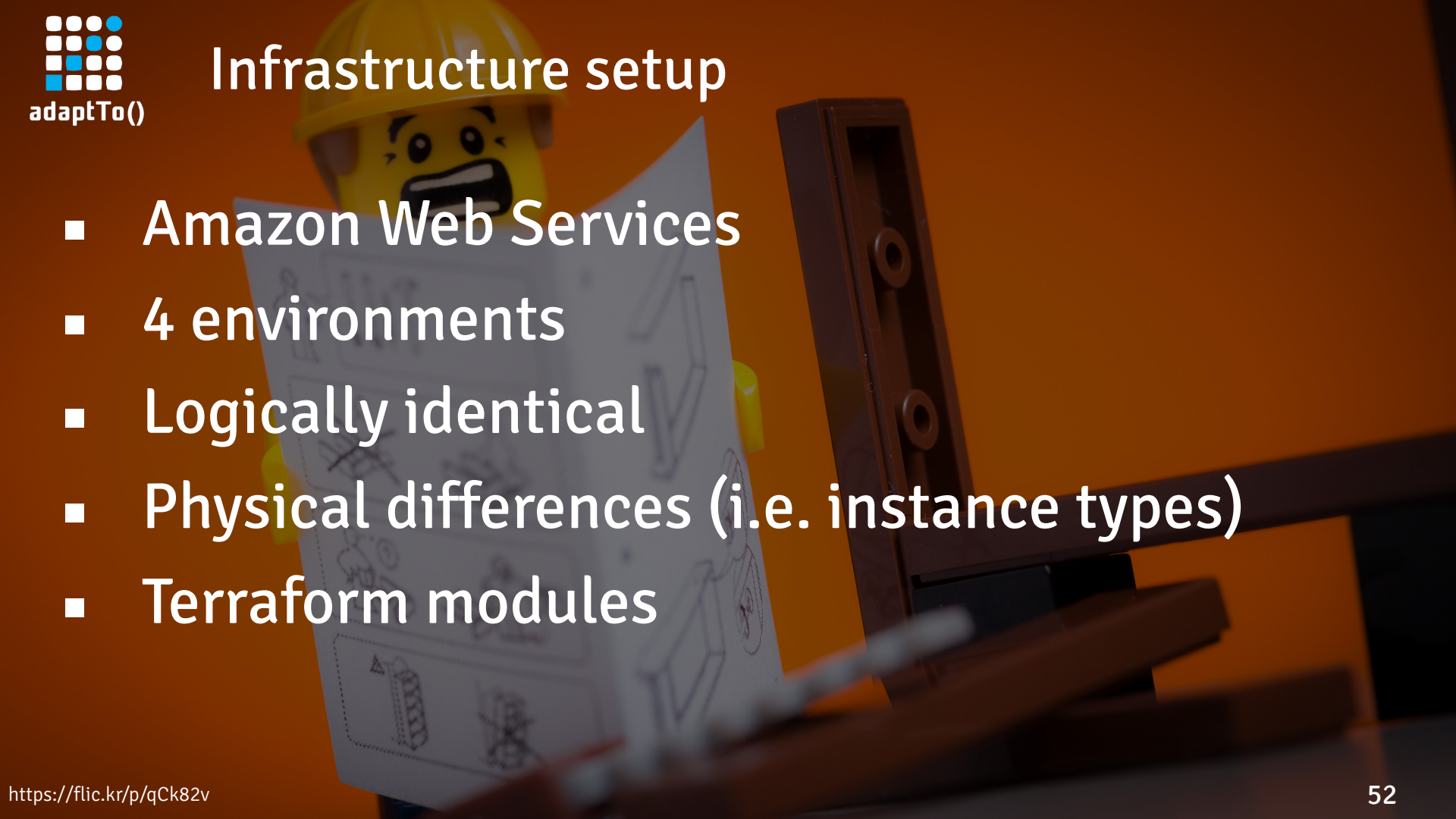
HashiCorp

Terraform

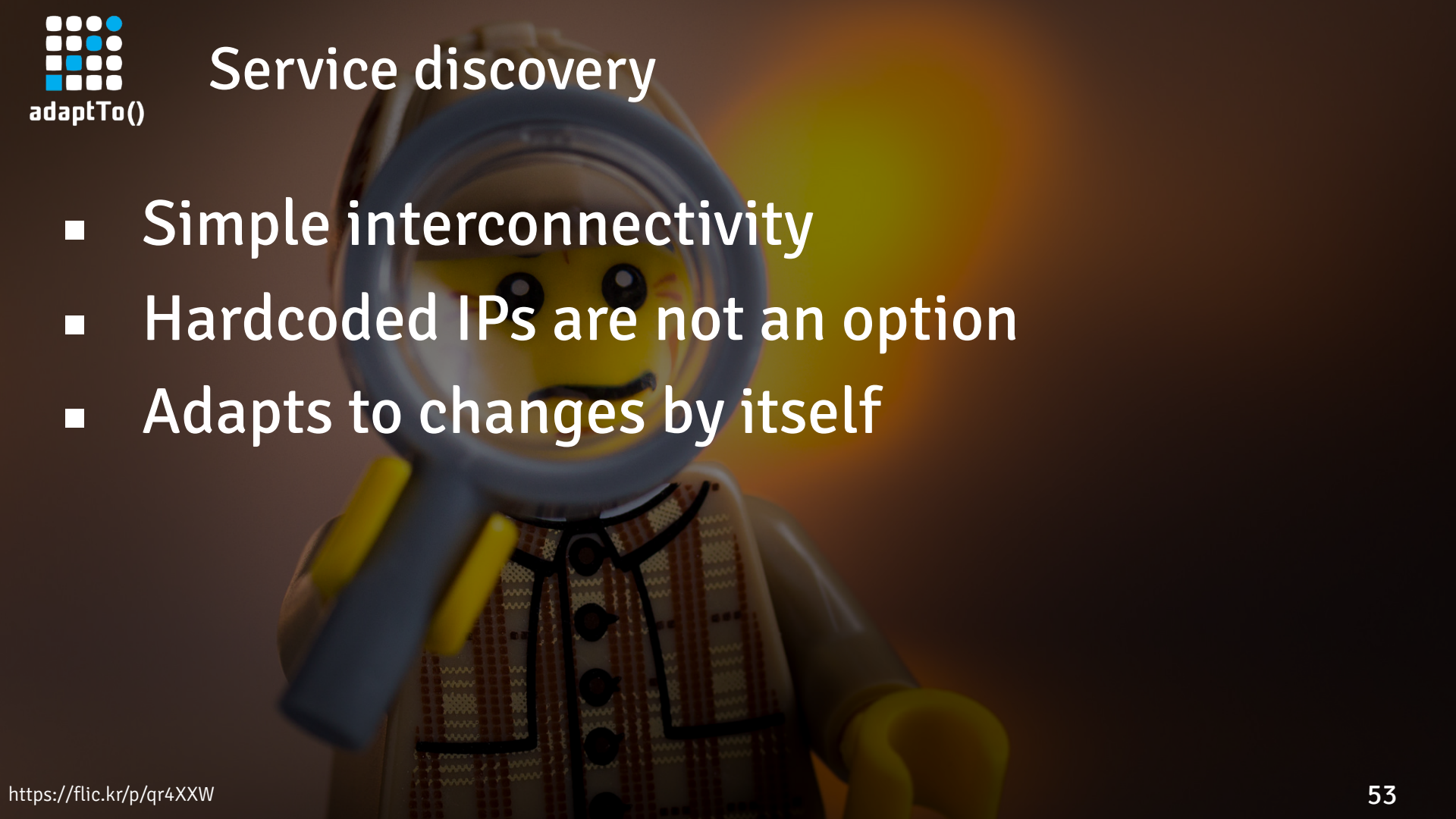
adaptTo()

# Infrastructure setup

- Amazon Web Services
- 4 environments
- Logically identical
- Physical differences (i.e. instance types)
- Terraform modules



https://flic.kr/p/qCk82v

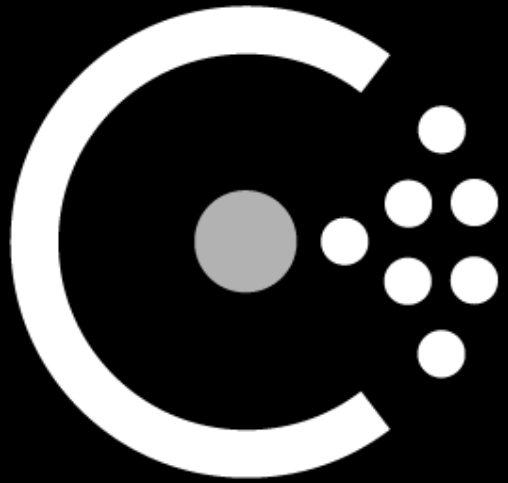


# Service discovery

- Simple interconnectivity
- Hardcoded IPs are not an option
- Adapts to changes by itself

https://flic.kr/p/qr4XXW

# Solution



HashiCorp

Consul


# Consul features

- Service discovery
- Load balancing
- Health checks


# Consul talk

https://goo.gl/aZjKsR

adaptTo()

# Automated deployments

adaptTo()

# Solution

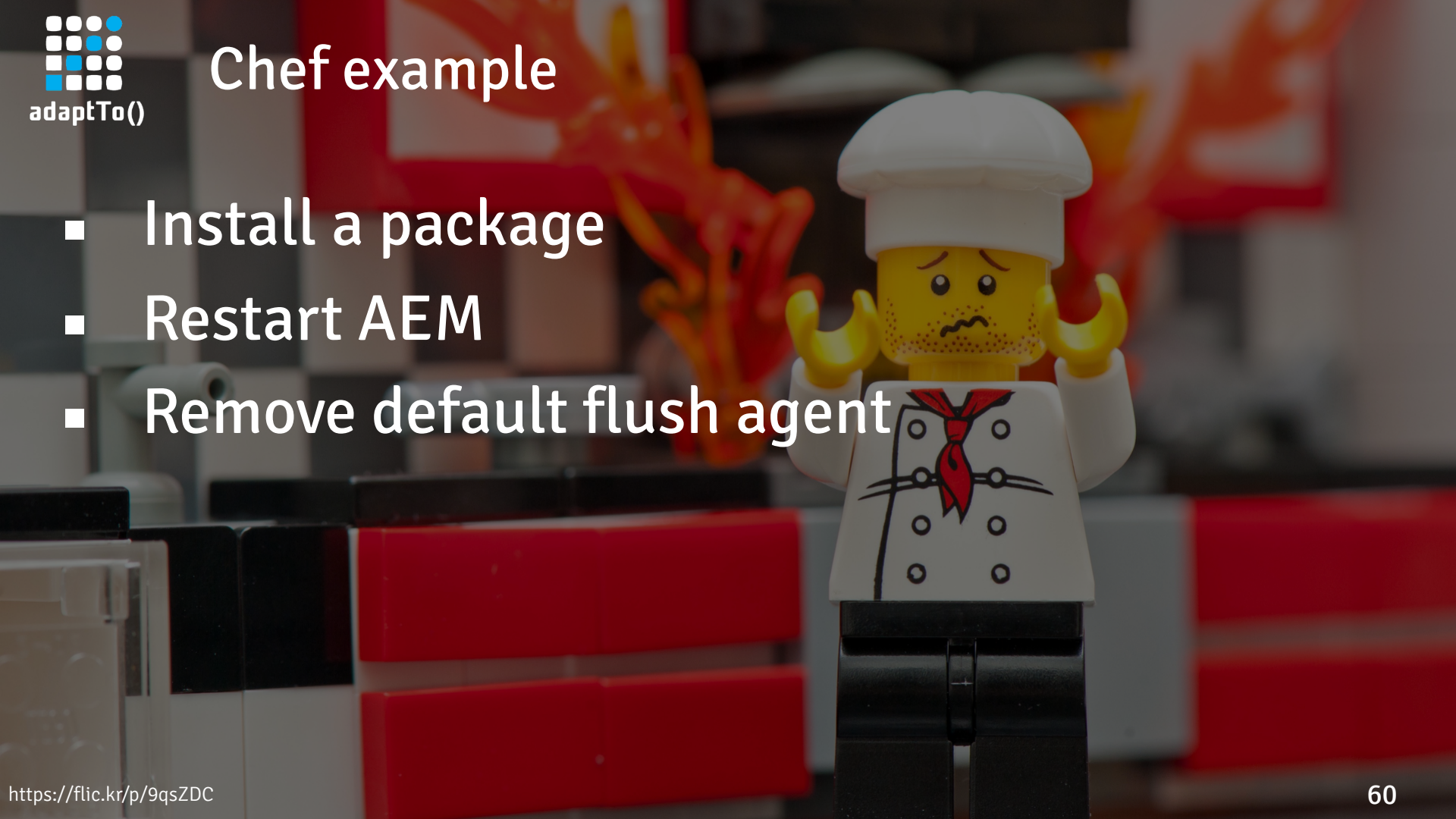
CHEF™

adaptTo()

# Chef

- Configuration management
- Each service has its own cookbook
- Deployment steps written in DSL

https://flic.kr/p/W25Lmv



# Chef example

- Install a package
- Restart AEM
- Remove default flush agent

https://flic.kr/p/9qsZDC

```
cq_package 'Author: Core app' do




end
```

```
cq_package 'Author: Core app' do
  username 'admin'
  password 'admin'
  instance 'http://localhost:4502'



end
```

```
cq_package 'Author: Core app' do
  username 'admin'
  password 'admin'
  instance 'http://localhost:4502'
  source 'https://artifacts.example.com/xyz/1.0/xyz-1.0.3-full.zip'
  http_user 'basic_auth_user'
  http_pass 'basic_auth_password'


end
```

```
cq_package 'Author: Core app' do
  username 'admin'
  password 'admin'
  instance 'http://localhost:4502'
  source 'https://artifacts.example.com/xyz/1.0/xyz-1.0.3-full.zip'
  http_user 'basic_auth_user'
  http_pass 'basic_auth_password'

  action :deploy
end
```

```
cq_package 'Author: Core app' do
  username 'admin'
  password 'admin'
  instance 'http://localhost:4502'
  source 'https://artifacts.example.com/xyz/1.0/xyz-1.0.3-full.zip'
  http_user 'basic_auth_user'
  http_pass 'basic_auth_password'

  action :deploy


end
```

```
cq_package 'Author: Core app' do
  username 'admin'
  password 'admin'
  instance 'http://localhost:4502'
  source 'https://artifacts.example.com/xyz/1.0/xyz-1.0.3-full.zip'
  http_user 'basic_auth_user'
  http_pass 'basic_auth_password'

  action :deploy


end

service 'cq62-author' do

end
```

```
cq_package 'Author: Core app' do
  username 'admin'
  password 'admin'
  instance 'http://localhost:4502'
  source 'https://artifacts.example.com/xyz/1.0/xyz-1.0.3-full.zip'
  http_user 'basic_auth_user'
  http_pass 'basic_auth_password'

  action :deploy


end

service 'cq62-author' do
  action :nothing
end
```

```
cq_package 'Author: Core app' do
  username 'admin'
  password 'admin'
  instance 'http://localhost:4502'
  source 'https://artifacts.example.com/xyz/1.0/xyz-1.0.3-full.zip'
  http_user 'basic_auth_user'
  http_pass 'basic_auth_password'

  action :deploy

  notifies :restart, 'service[cq62-author]', :immediately
end

service 'cq62-author' do
  action :nothing
end
```

```
cq_jcr 'Author: /etc/replication/agents.author/flush' do



end
```

```
cq_jcr 'Author: /etc/replication/agents.author/flush' do
  path '/etc/replication/agents.author/flush'




end
```

```
cq_jcr 'Author: /etc/replication/agents.author/flush' do
  path '/etc/replication/agents.author/flush'
  username 'admin'
  password 'admin'
  instance 'http://localhost:4502'


end
```

```
cq_jcr 'Author: /etc/replication/agents.author/flush' do
  path '/etc/replication/agents.author/flush'
  username 'admin'
  password 'admin'
  instance 'http://localhost:4502'

  action :delete
end
```

# What's currently available?

- cq_package
- cq_osgi_config
- cq_osgi_bundle
- cq_osgi_component
- cq_user
- cq_jcr
- groovy_script
- apm_script
- …

https://flic.kr/p/qhRjAF


# AEM cookbook

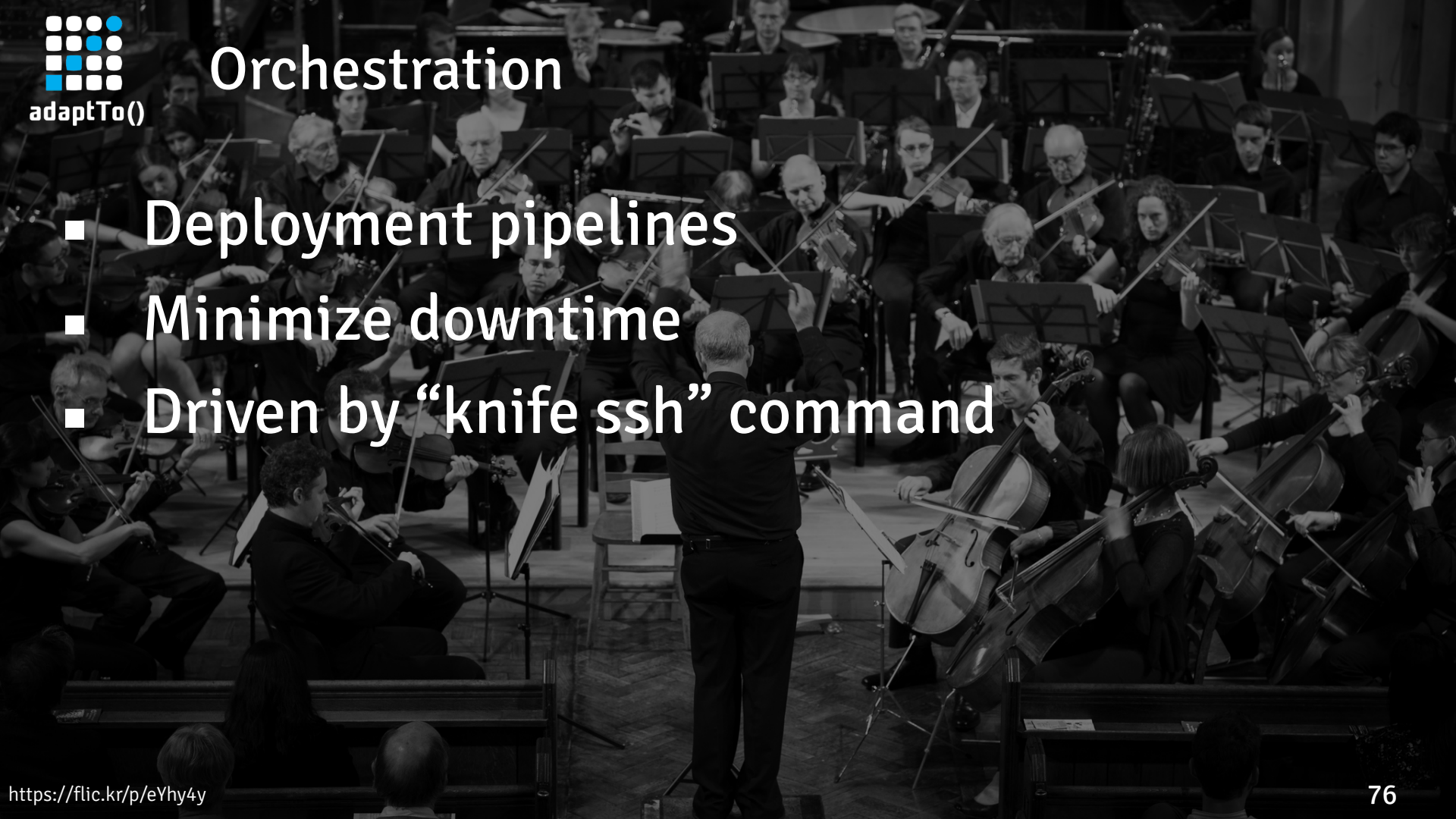
github.com/jwadolowski/cookbook-cq


# Orchestration

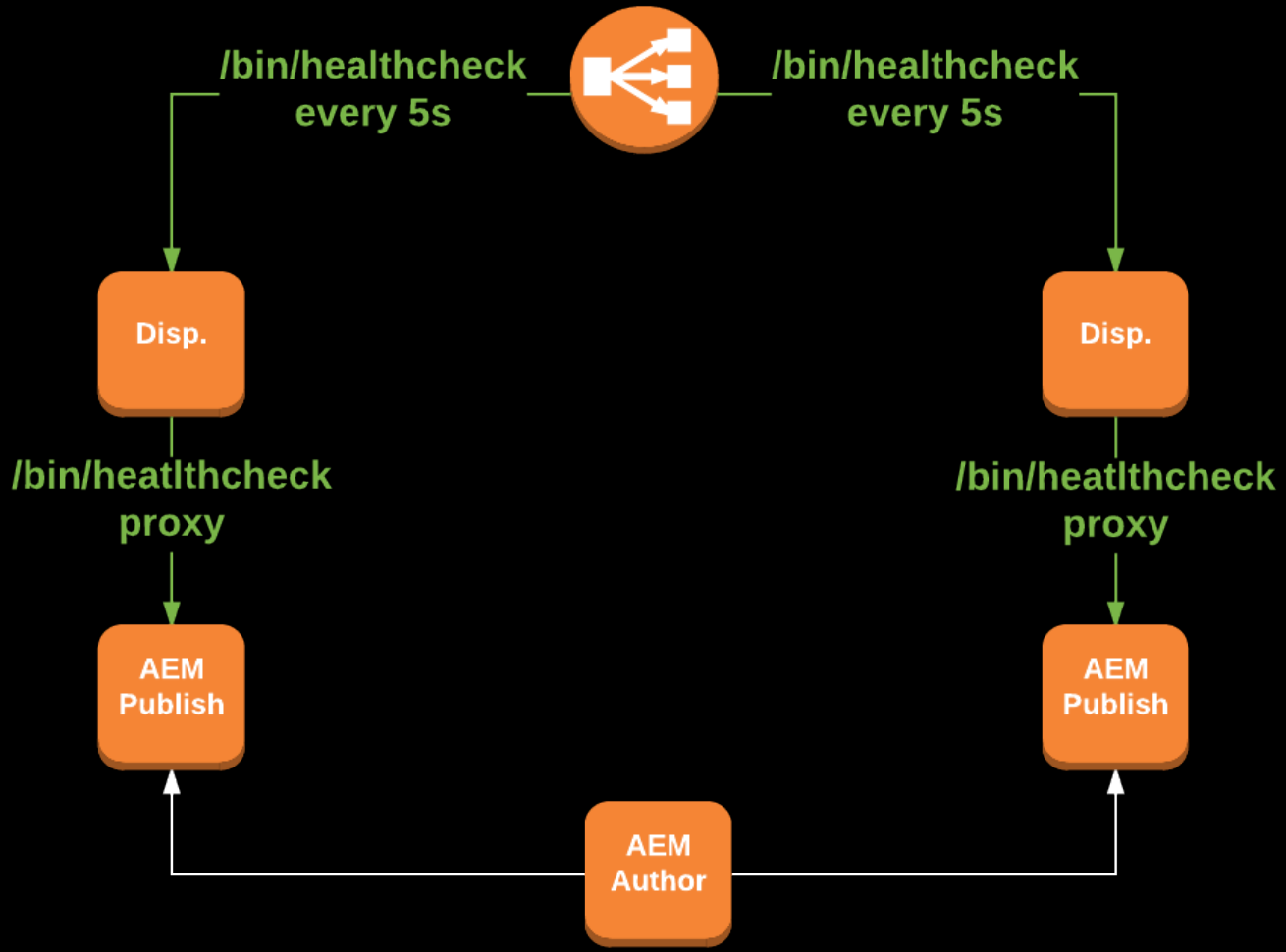
adaptTo()

# Orchestration

- Deployment pipelines
- Minimize downtime
- Driven by “knife ssh” command



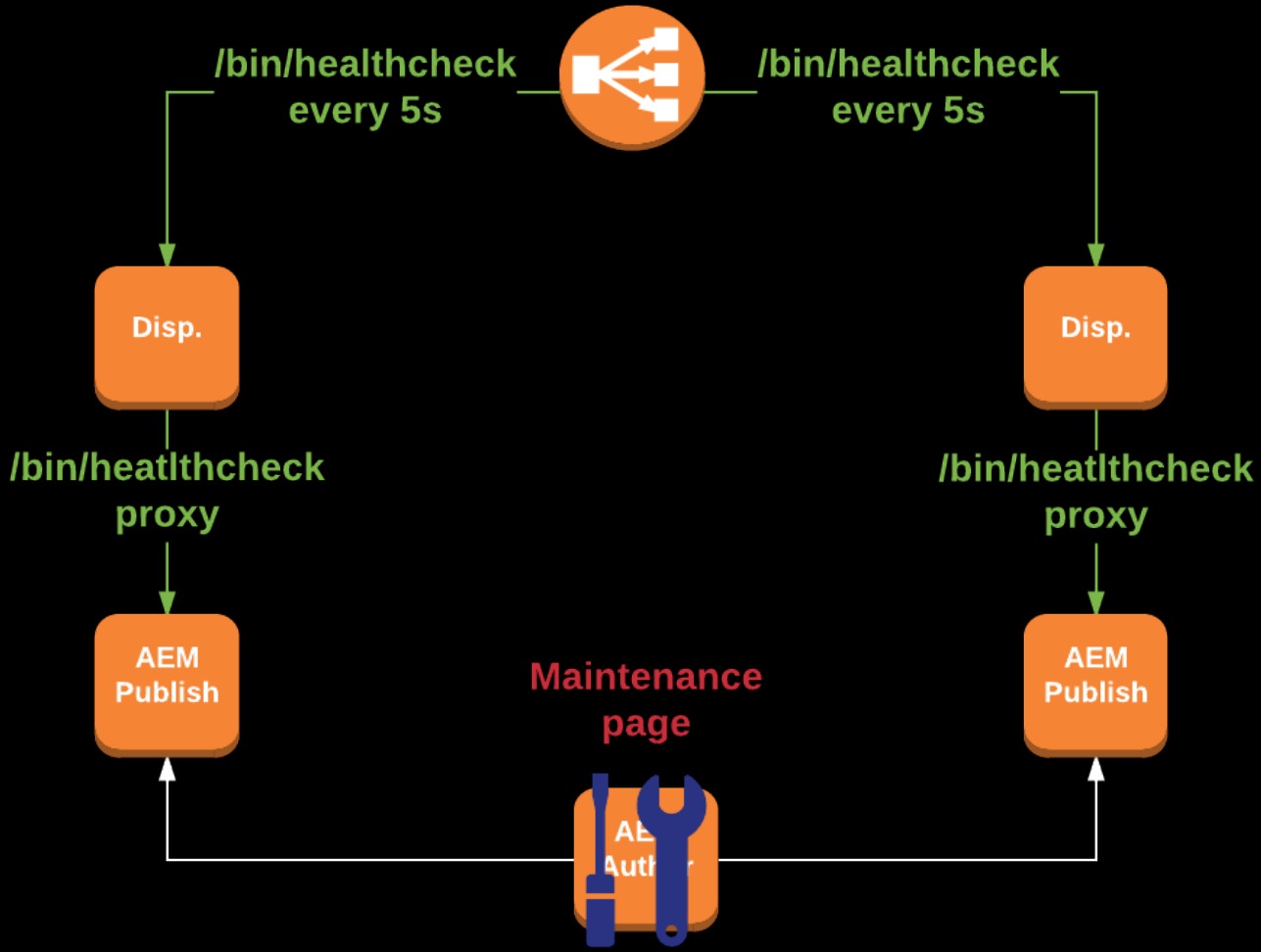
https://flic.kr/p/eYhy4y


/bin/healthcheck
every 5s
/bin/healthcheck
every 5s
Disp.
Disp.
/bin/heatlthcheck
proxy
/bin/heatlthcheck
proxy
AEM
Publish
AEM
Publish
AEM
Author

```
$ knife ssh
```

```
$ knife ssh \
    'chef_environment:xyz-prod AND tags:aem AND tags:author'
```

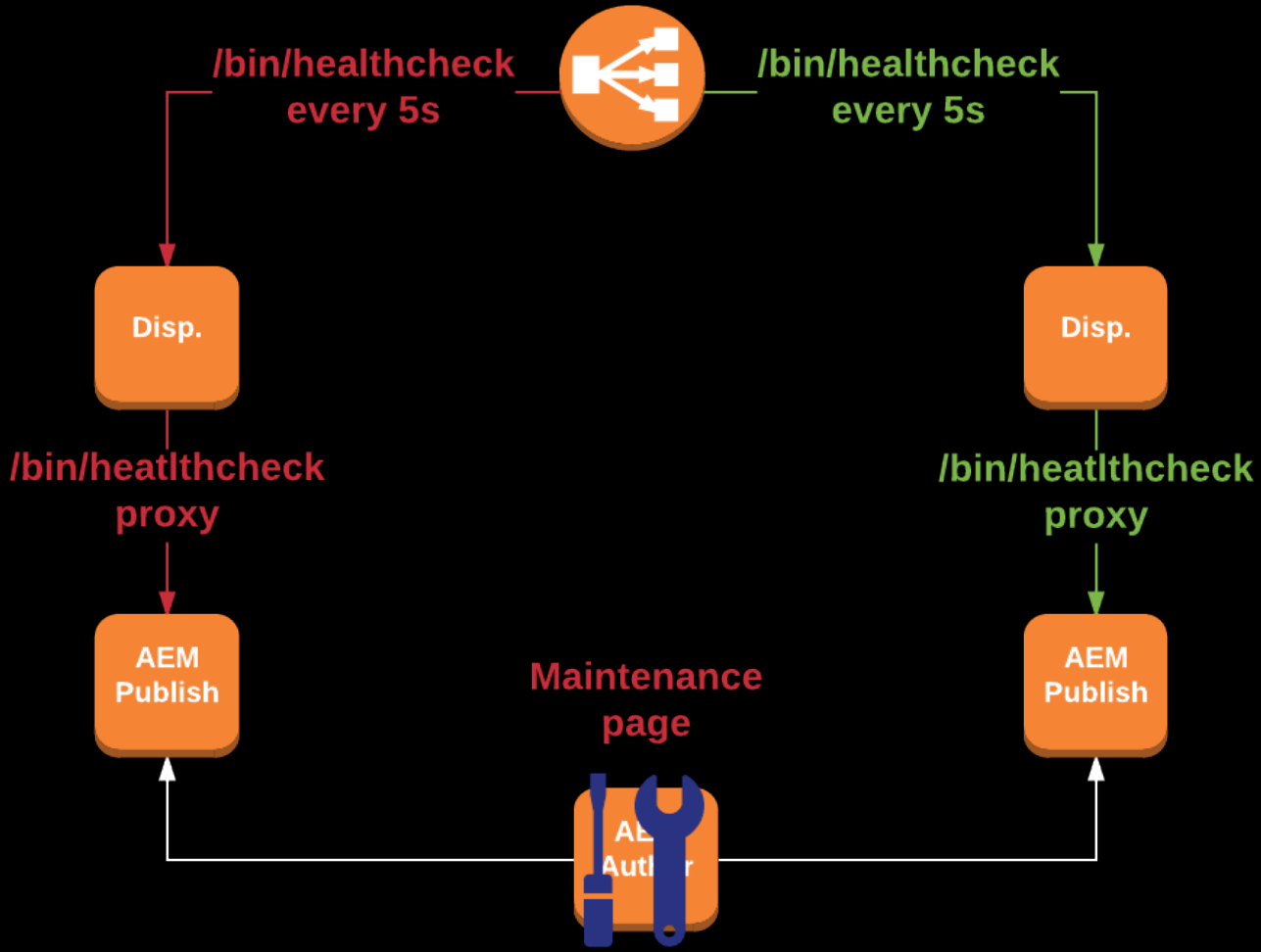
```
$ knife ssh \
    'chef_environment:xyz-prod AND tags:aem AND tags:author' \
    'sudo -u apache /bin/touch /path/to/maintenance/maintenance.lock'
```


/bin/healthcheck
every 5s
/bin/healthcheck
every 5s
Disp.
Disp.
/bin/heatlthcheck
proxy
/bin/heatlthcheck
proxy
AEM
Publish
AEM
Publish
Maintenance
page

```
$ knife ssh
```

```
$ knife ssh \
    'chef_environment:xyz-prod AND tags:dispatcher AND tags:us-east-1a'
```

```
$ knife ssh \
    'chef_environment:xyz-prod AND tags:dispatcher AND tags:us-east-1a' \
    'sudo -u apache /bin/touch /path/to/maintenance/deployment.lock'
```

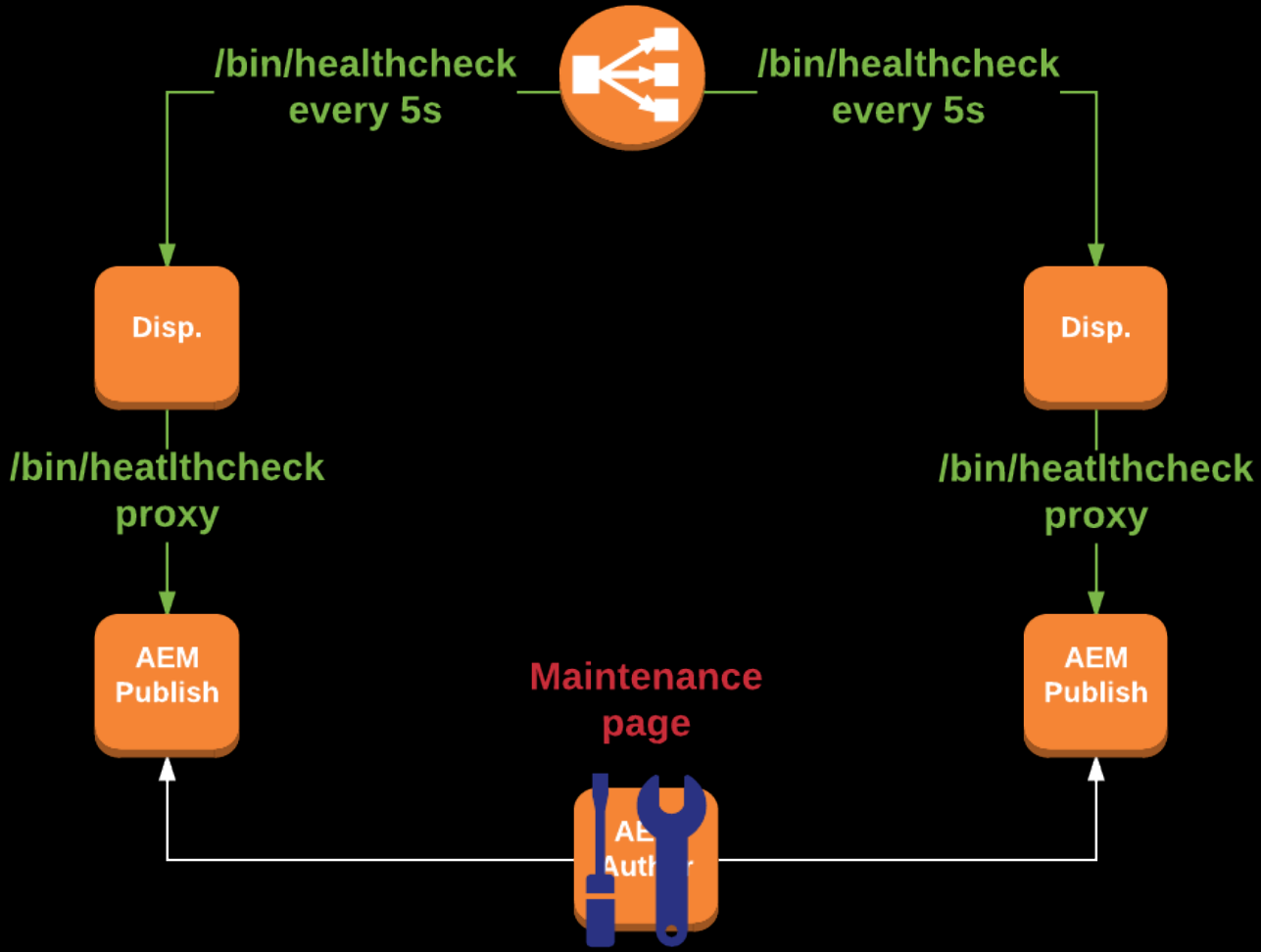

/bin/healthcheck
every 5s
/bin/healthcheck
every 5s
Disp.
Disp.
/bin/heatlthcheck
proxy
/bin/heatlthcheck
proxy
AEM
Publish
AEM
Publish
Maintenance
page

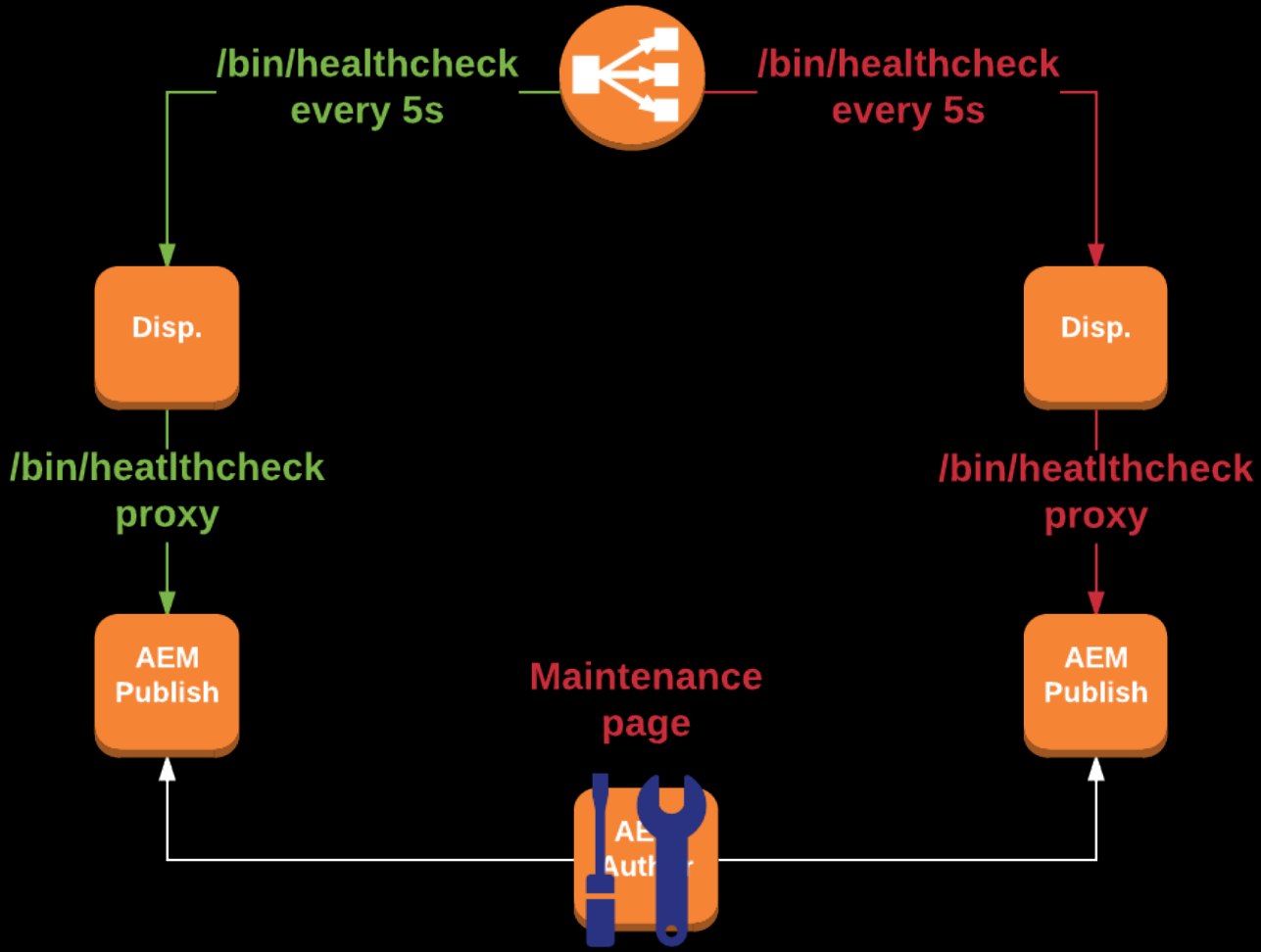
```
$ knife ssh
```

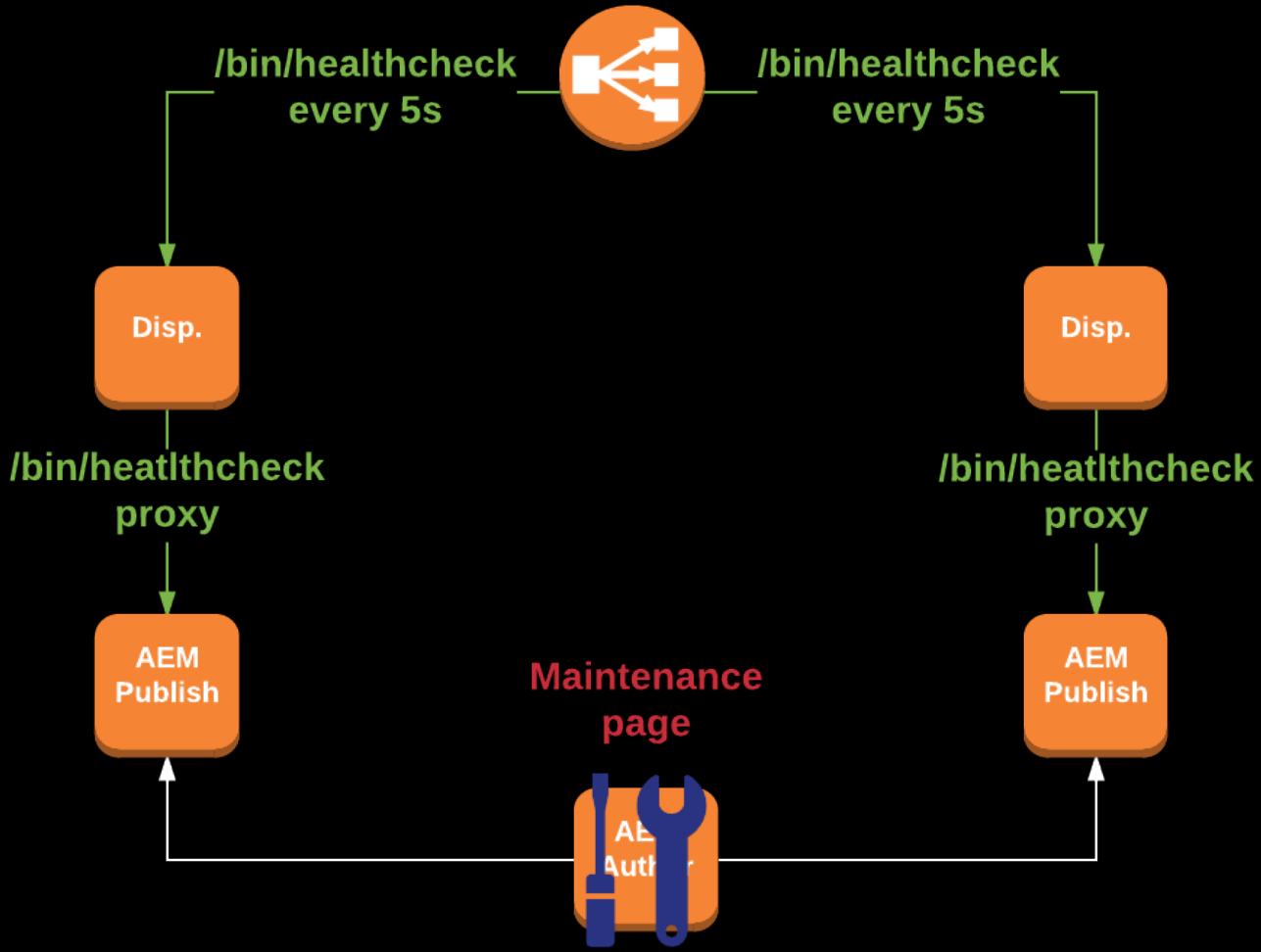
```
$ knife ssh \
    'chef_environment:xyz-prod AND (
        (tags:dispatcher AND tags:us-east-1a) OR
        (tags:aem AND tags:publish AND tags:us-east-1a)
    )'
```

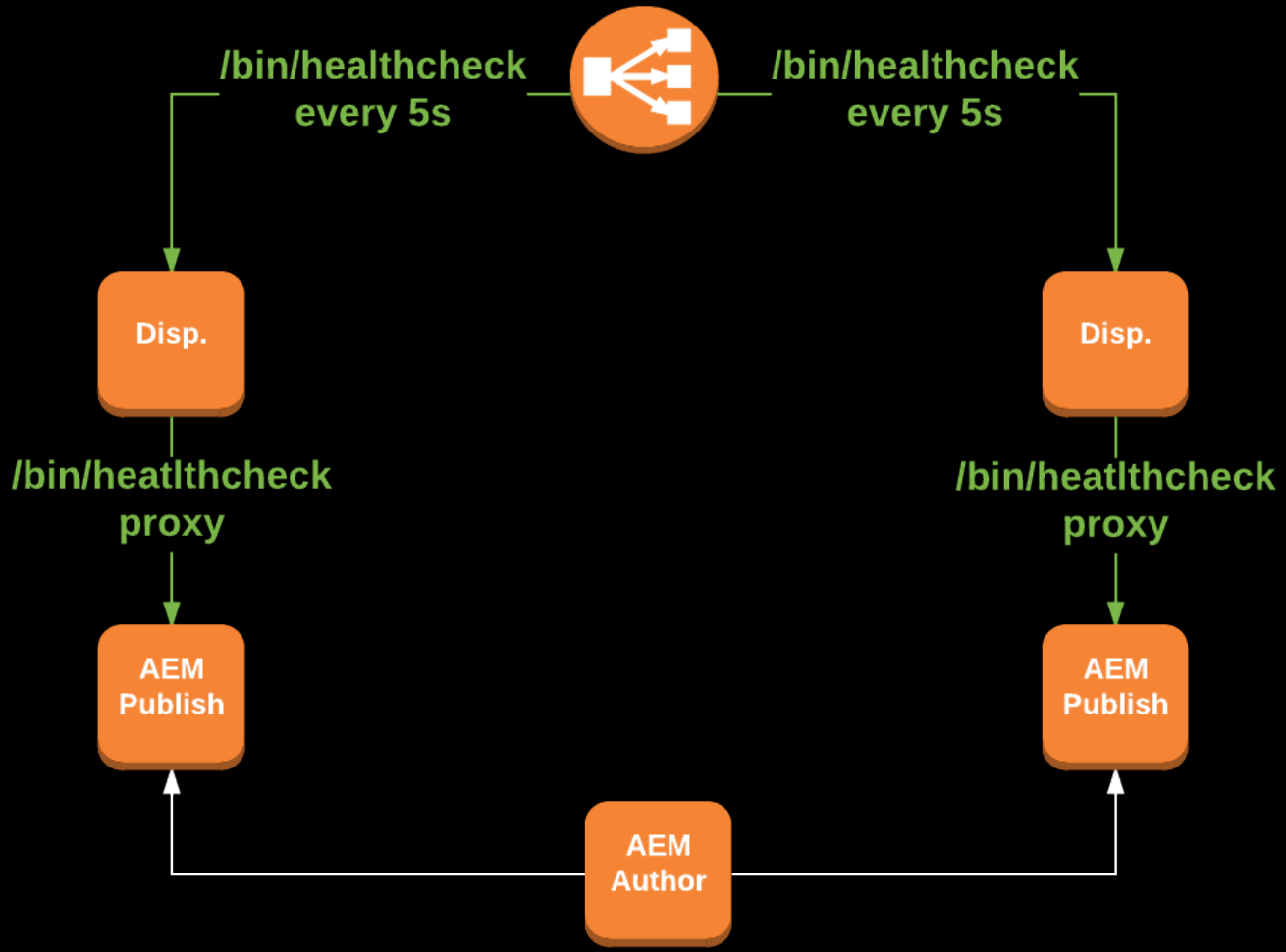
```
$ knife ssh \
    'chef_environment:xyz-prod AND (
        (tags:dispatcher AND tags:us-east-1a) OR
        (tags:aem AND tags:publish AND tags:us-east-1a)
    )' \
    'sudo /usr/bin/chef-client'
```

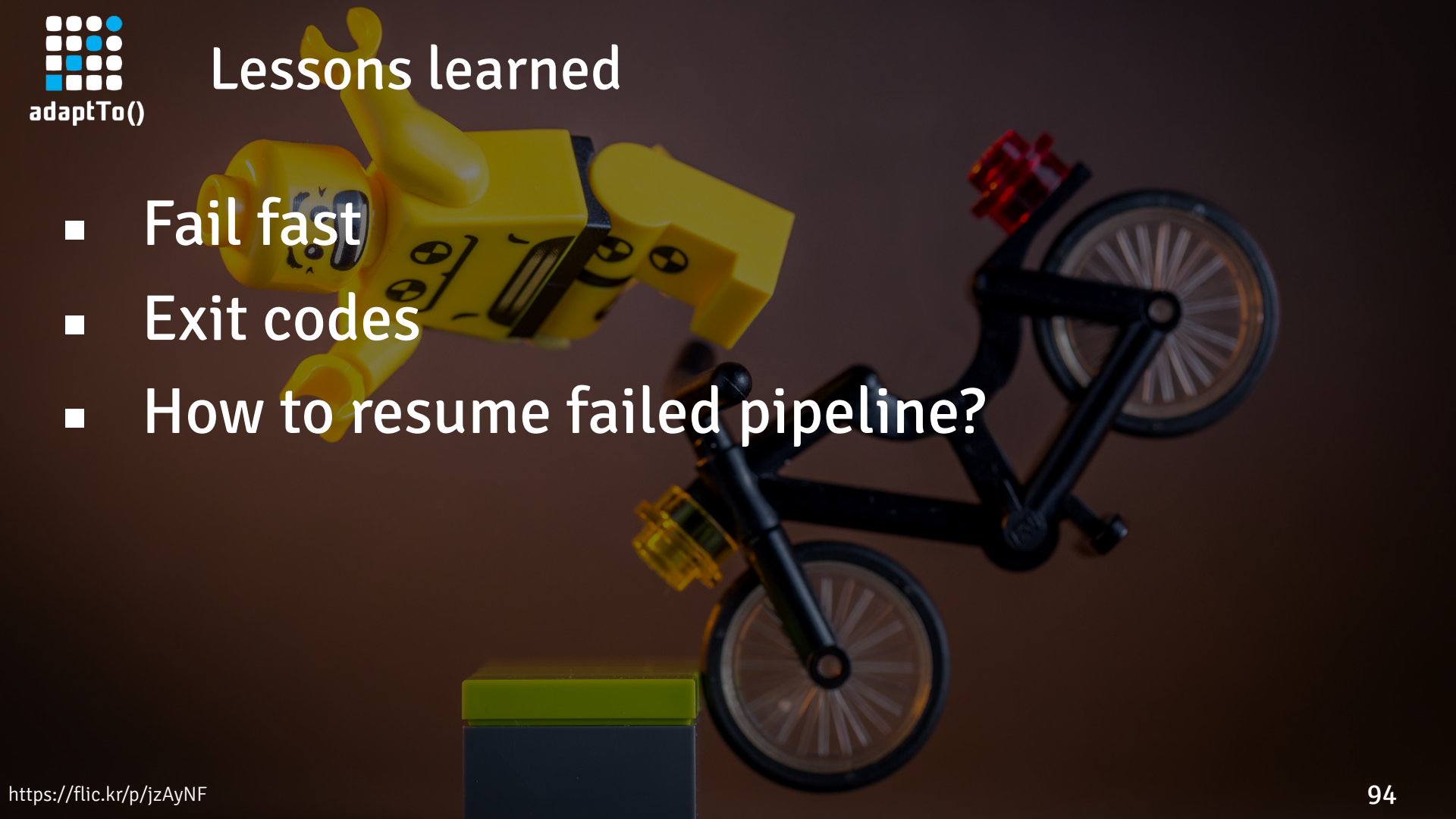
```
$ knife ssh \
    'chef_environment:xyz-prod AND tags:dispatcher AND tags:us-east-1a' \
    'sudo -u apache /usr/bin/rm -f /path/to/maintenance/deployment.lock'
```


/bin/healthcheck
every 5s
/bin/healthcheck
every 5s
Disp.
Disp.
/bin/heatlthcheck
proxy
/bin/heatlthcheck
proxy
AEM
Publish
AEM
Publish
Maintenance
page


/bin/healthcheck
every 5s
/bin/healthcheck
every 5s
Disp.
Disp.
/bin/heatlthcheck
proxy
/bin/heatlthcheck
proxy
AEM
Publish
AEM
Publish
Maintenance
page


/bin/healthcheck
every 5s
/bin/healthcheck
every 5s
Disp.
Disp.
/bin/heatlthcheck
proxy
/bin/heatlthcheck
proxy
AEM
Publish
AEM
Publish
Maintenance
page


/bin/healthcheck
every 5s
/bin/healthcheck
every 5s
Disp.
Disp.
/bin/heatlthcheck
proxy
/bin/heatlthcheck
proxy
AEM
Publish
AEM
Publish
AEM
Author



# Lessons learned

- Fail fast
- Exit codes
- How to resume failed pipeline?

https://flic.kr/p/jzAyNF


# Danke schön!