KUBERNETES AND TAXES

Lessons learned at The Norwegian Tax Administration(NTA)

Bjarte Karlsen
@bjartek

THE AURORA PLATFORM

Faster development and more efficient ops

- PaaS built upon OpenShift
- At the core of the plattform is the declerative config format AuroraConfig and the AuroraAPI.
- The AuroraAPI supports deploying applications and observing their status while running.
- Building applications from source is done in the AuroraPipeline that uses a central build logic.

Why did you make things inhouse?

- Both Kubernetes and OpenShift lack a concept we have called **affiliation**. That is groups of people that can administer or view objects for several projects. We have several different development teams that work on our clusters and we want them to be able to work in **isolation**.
- The ability to deploy applications to **several clusters** in one command is highly desired within our organization. Our network infrastructure implies that we need to have multiple clusters.
- [illegible] projects we want to avoid duplication. Our **declarative** config format AuroraConfig supports **composition** with **sane defaults**

https://skatteetaten.github.io/aurora

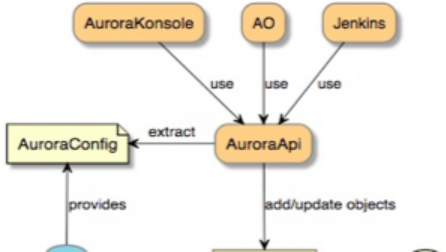
In order to avoid 'wall-of-yaml' we use a declarative, composable configuration format with sane defaults.

Bjarte Karlsen, Technical Architect NTA

How do we deploy

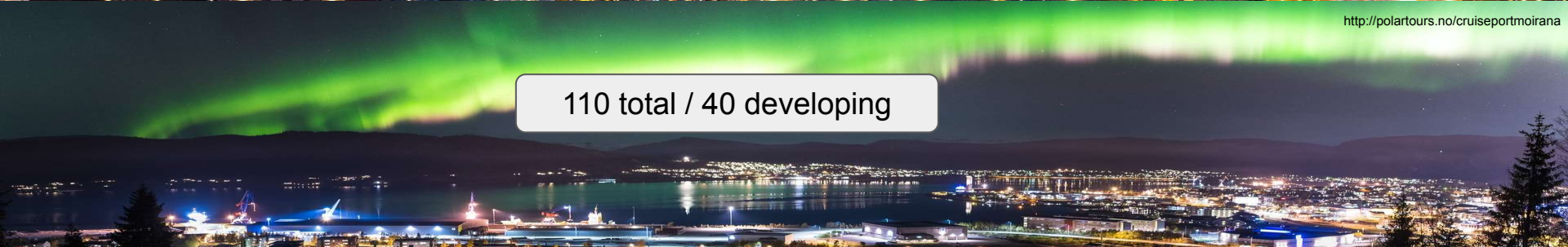
An deploy starts with triggering the AuroraAPI from either of the userfacing clients AO, AuroraKonsole or AuroraPipeline. The API will then extract and merge AuroraConfig in order to create a AuroraDeploymentSpec.

Synchrous integrations are run and the result of both are assembled into Kubernetes objects that are applied to the cluster. Async integrations will that use the




300 total / 110 developing
https://grimstadminby.no


600 total / 250 developing
https://www.independent.co.uk


http://polartours.no/cruiseportmoirana
110 total / 40 developing

Skatteetaten

“Faster development
and
more efficient ops”

Skatteetaten

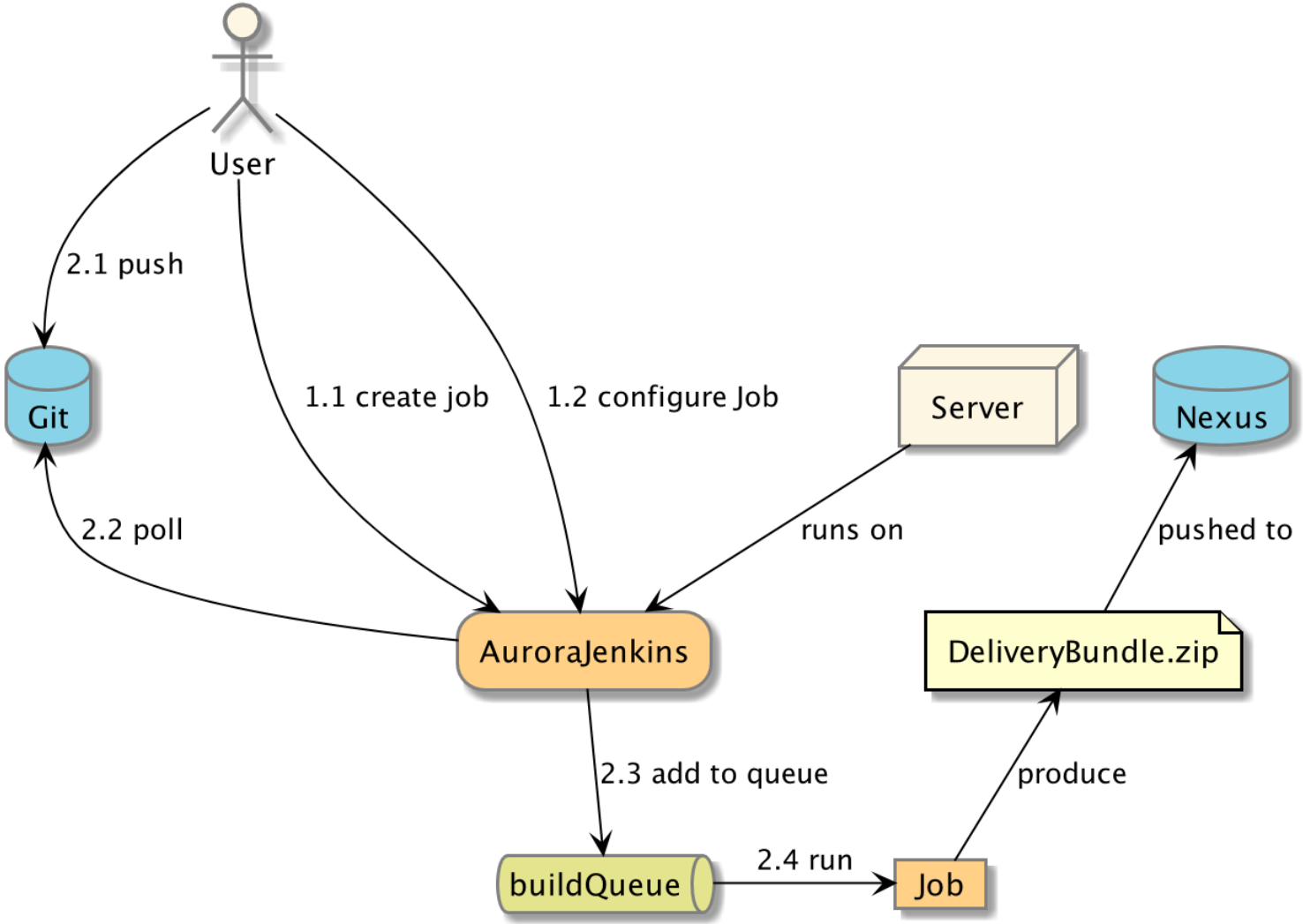
6 clusters

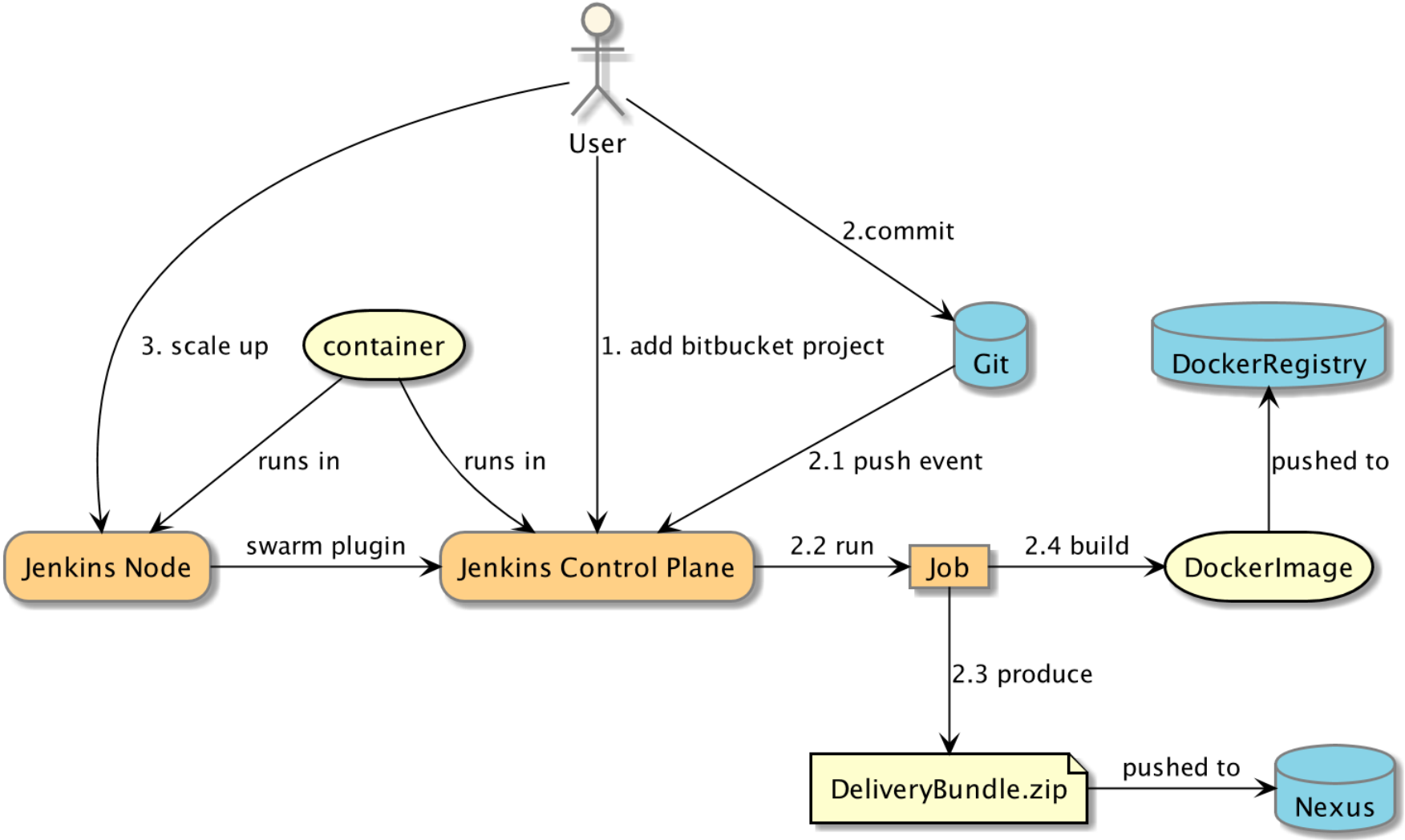
	Backend	DMZ
Prod	12 teams, 190 apps, 450 pods	5 teams, 20 apps, 70 pods
Test	17 teams, 250 apps, 1100 pods	5 teams, 25 apps, 100 pods
Dev	17 teams, 250 apps, 1500 pods	5 teams, 60 apps, 100 pods

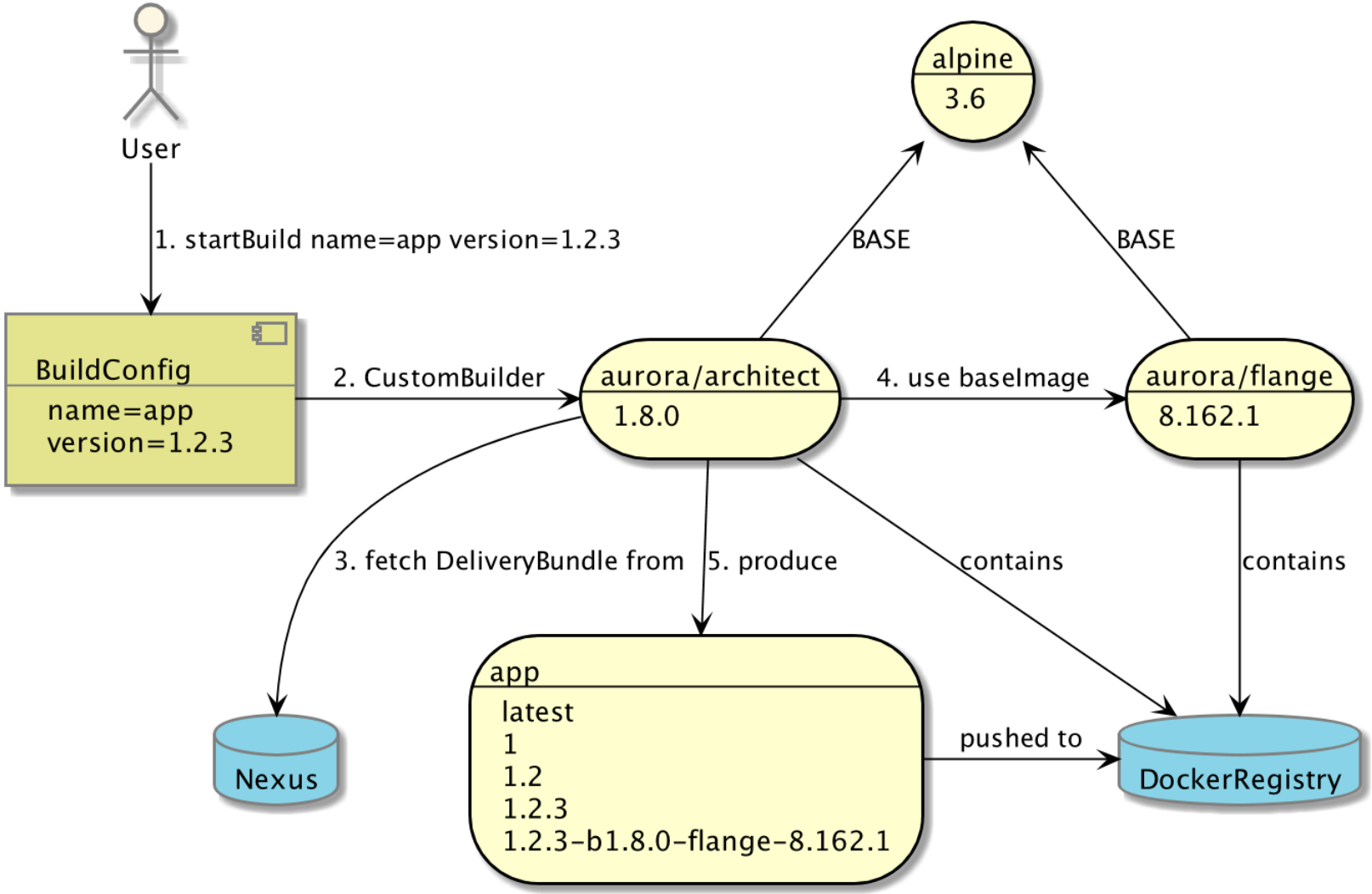
Agenda

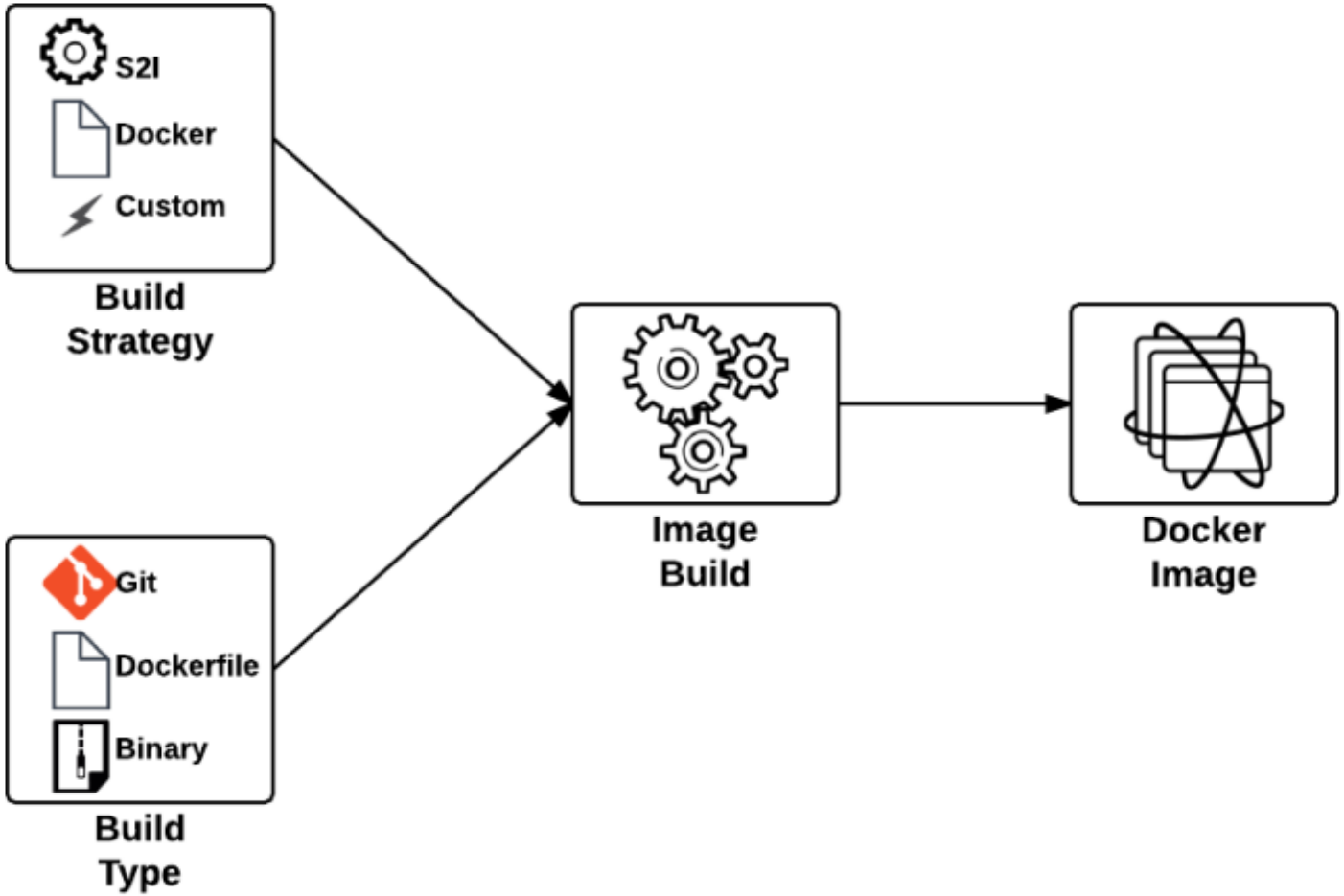
- Building
- Deploying
- External Integrations
- Running
- General lessons


BUILDING APPLICATIONS

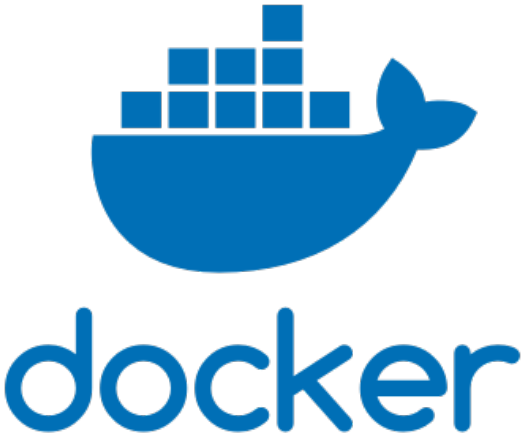

User
2.1 push
Git
1.1 create job
1.2 configure Job
Server
Nexus
2.2 poll
runs on
pushed to
AuroraJenkins
DeliveryBundle.zip
2.3 add to queue
produce
buildQueue
2.4 run
Job


User
2.commit
3. scale up
container
1. add bitbucket project
Git
DockerRegistry
runs in
runs in
2.1 push event
pushed to
Jenkins Node
swarm plugin
Jenkins Control Plane
2.2 run
Job
2.4 build
DockerImage
2.3 produce
DeliveryBundle.zip
pushed to
Nexus


User
1. startBuild name=app version=1.2.3
BuildConfig
name=app
version=1.2.3
2. CustomBuilder
aurora/architect
1.8.0
alpine
3.6
BASE
BASE
4. use baseImage
aurora/flange
8.162.1
3. fetch DeliveryBundle from
5. produce
contains
contains
Nexus
app
latest
1
1.2
1.2.3
1.2.3-b1.8.0-flange-8.162.1
pushed to
DockerRegistry


S2I
Docker
Custom
Build Strategy
Git
Dockerfile
Binary
Build Type
Image Build
Docker Image

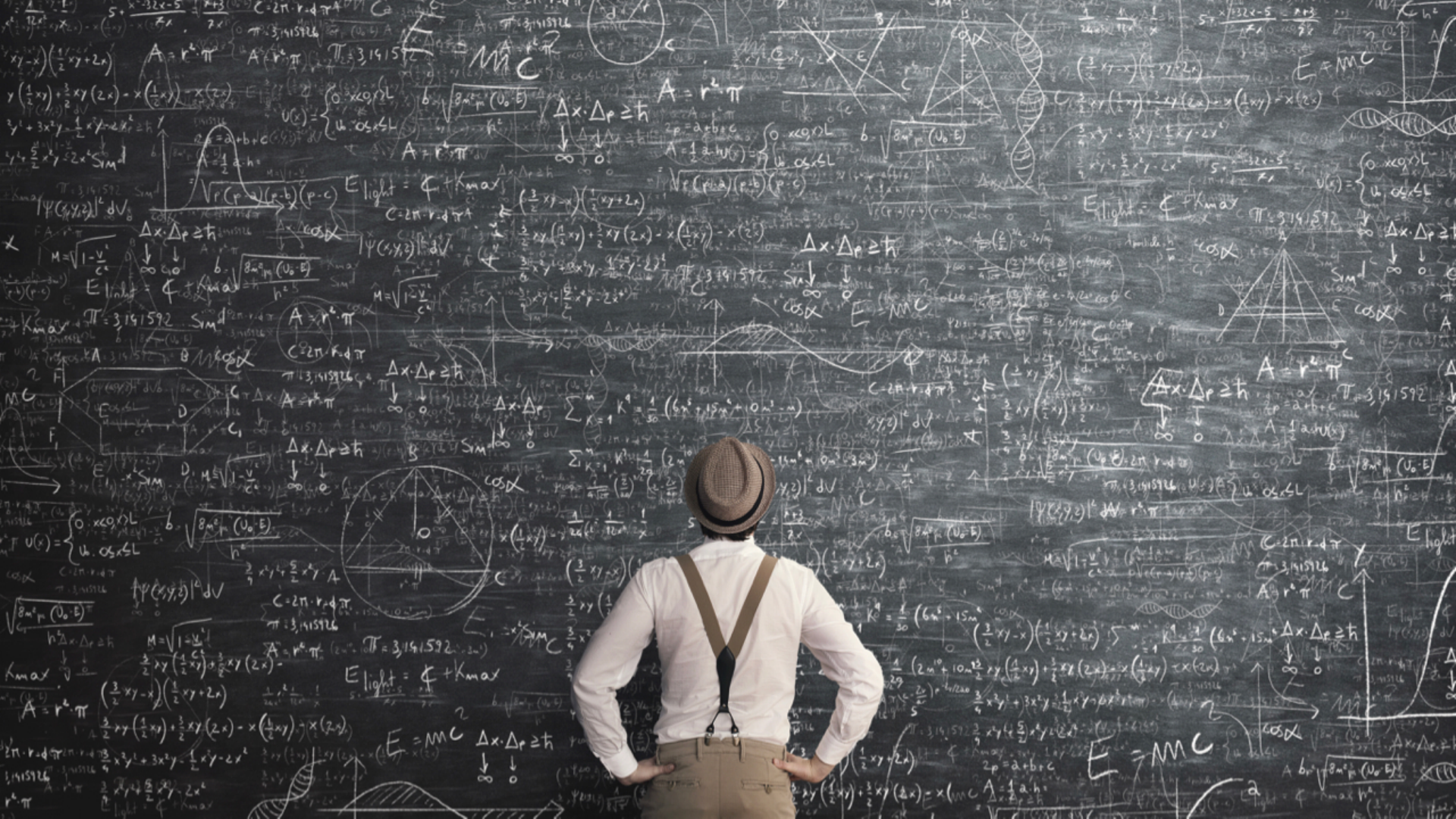

Java™


docker


git push

Atlassian

Bitbucket

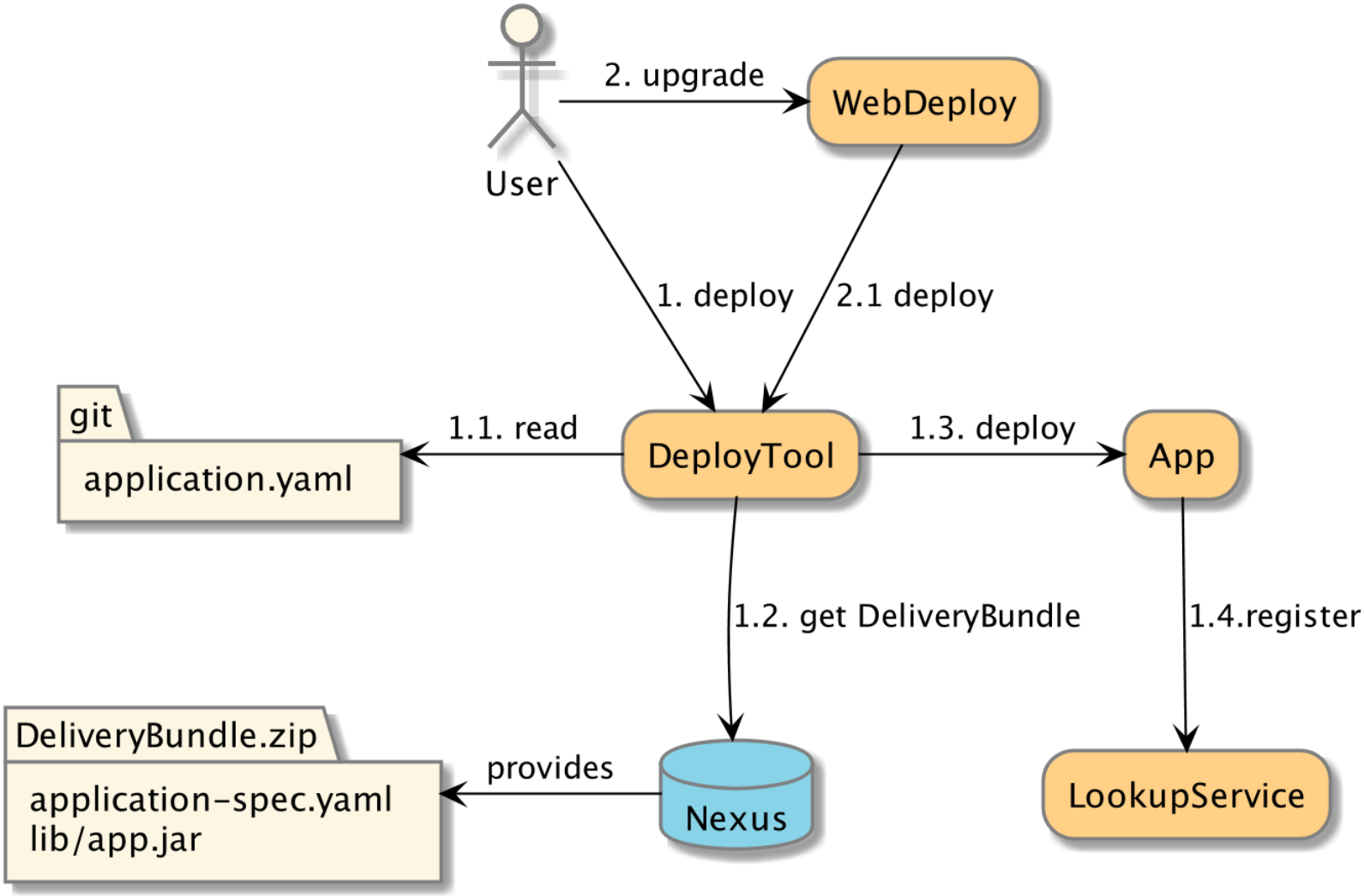


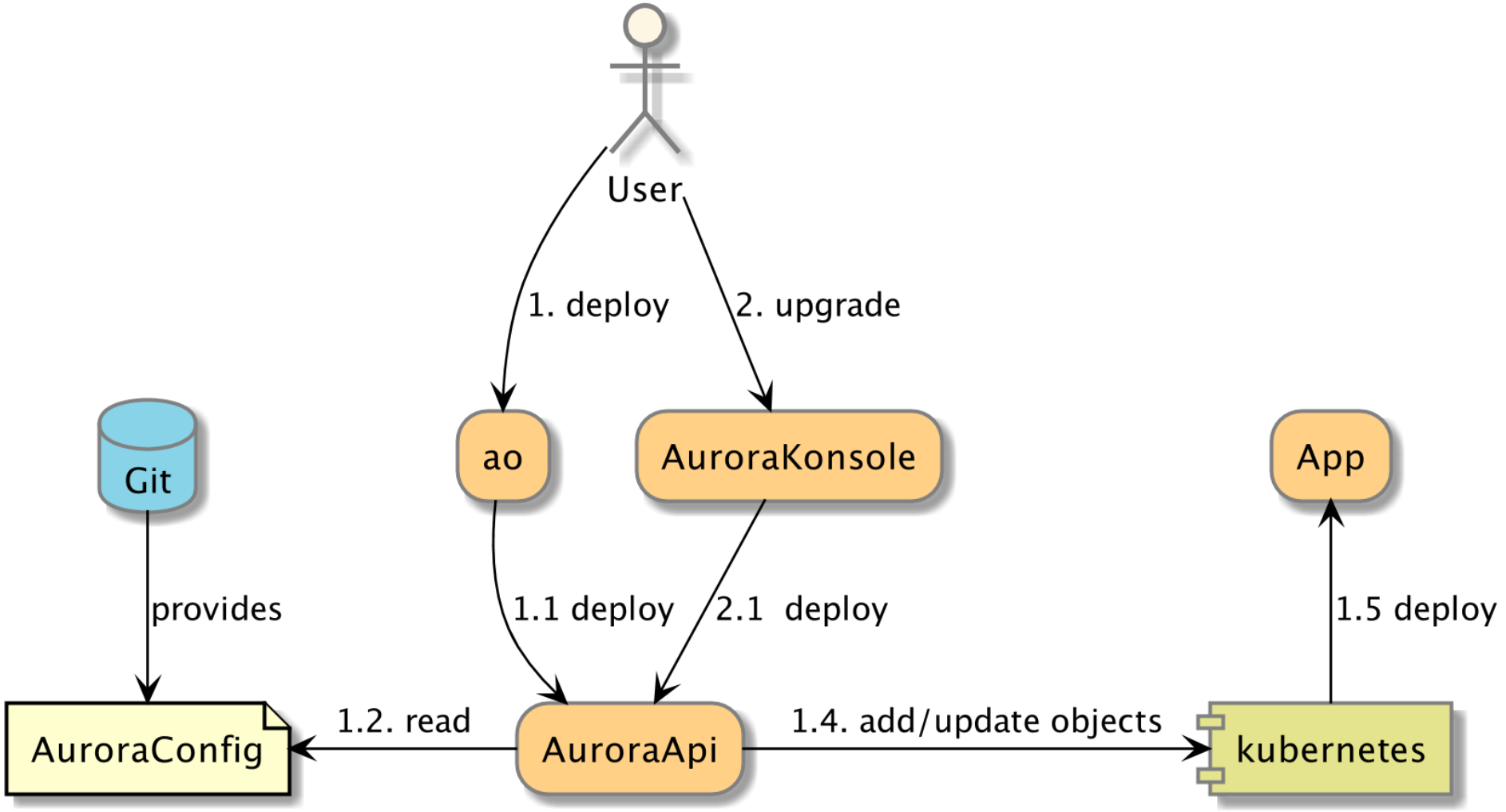
Skatteetaten

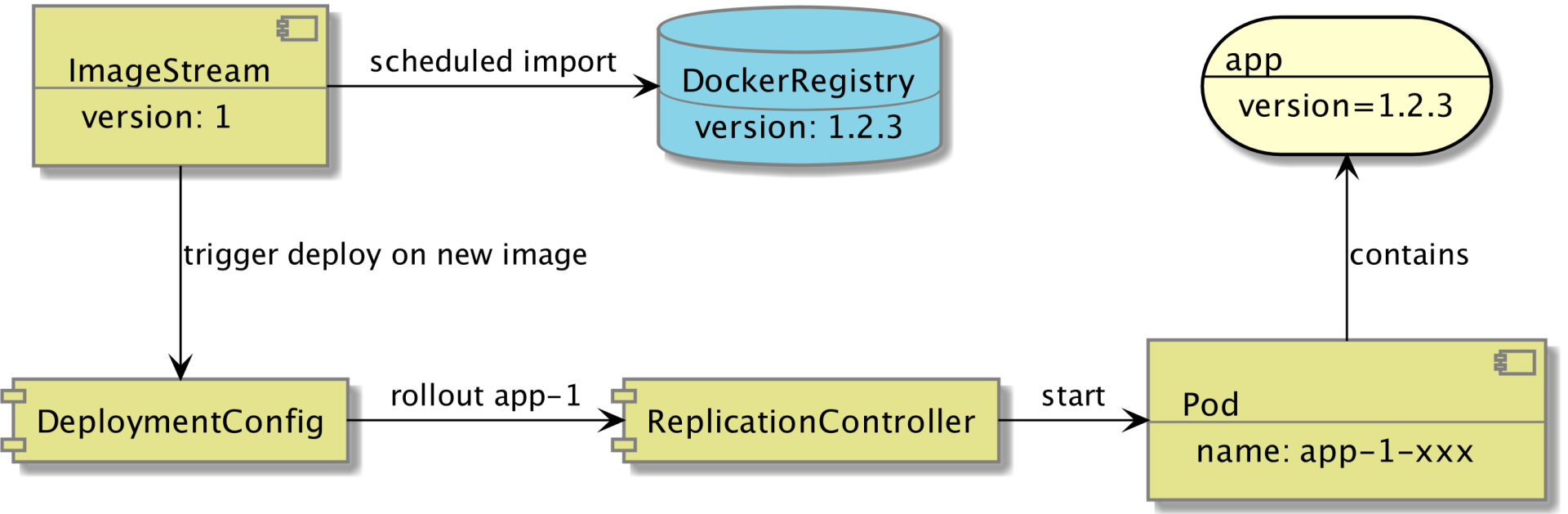
AURORA BUILD CONTRACT

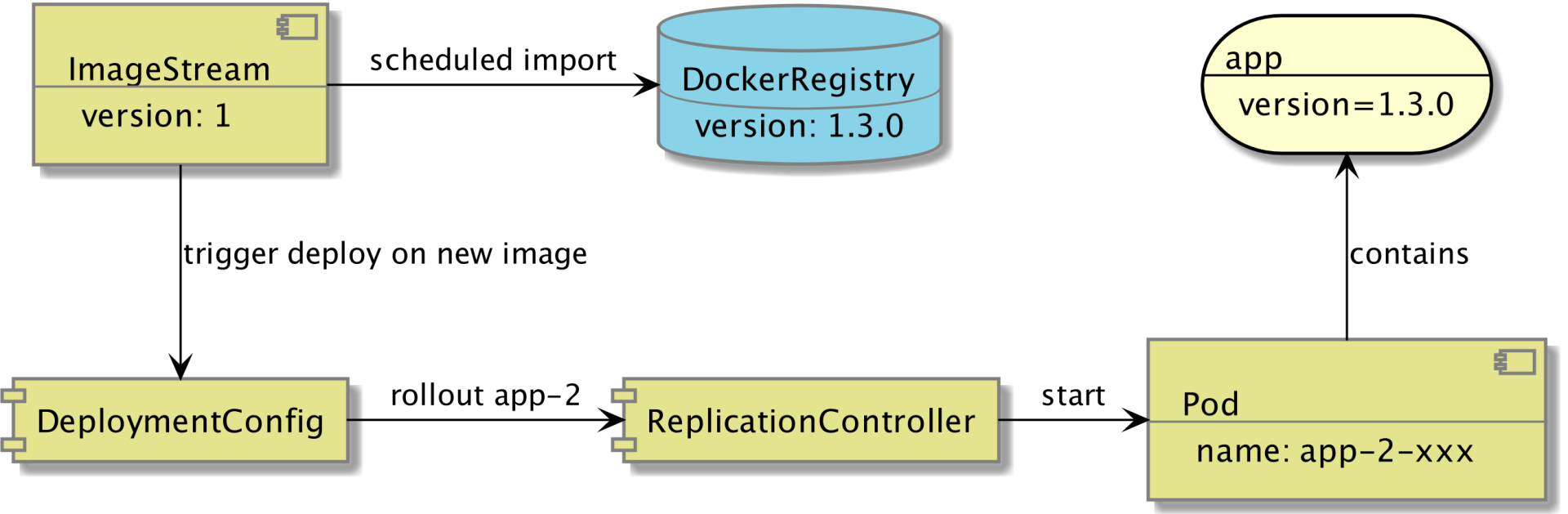
- Jenkinsfile
- produce DeliveryBundle to Nexus
  - openshift/metadata.json
    - mainClass
    - JVM args
    - app properties
  - lib/app.jar (not fat-jar, make jar not war)
  - lib/dep1, dep2, dep3 aso
- Semantic versioning
- produce DockerImage via BuildConfig

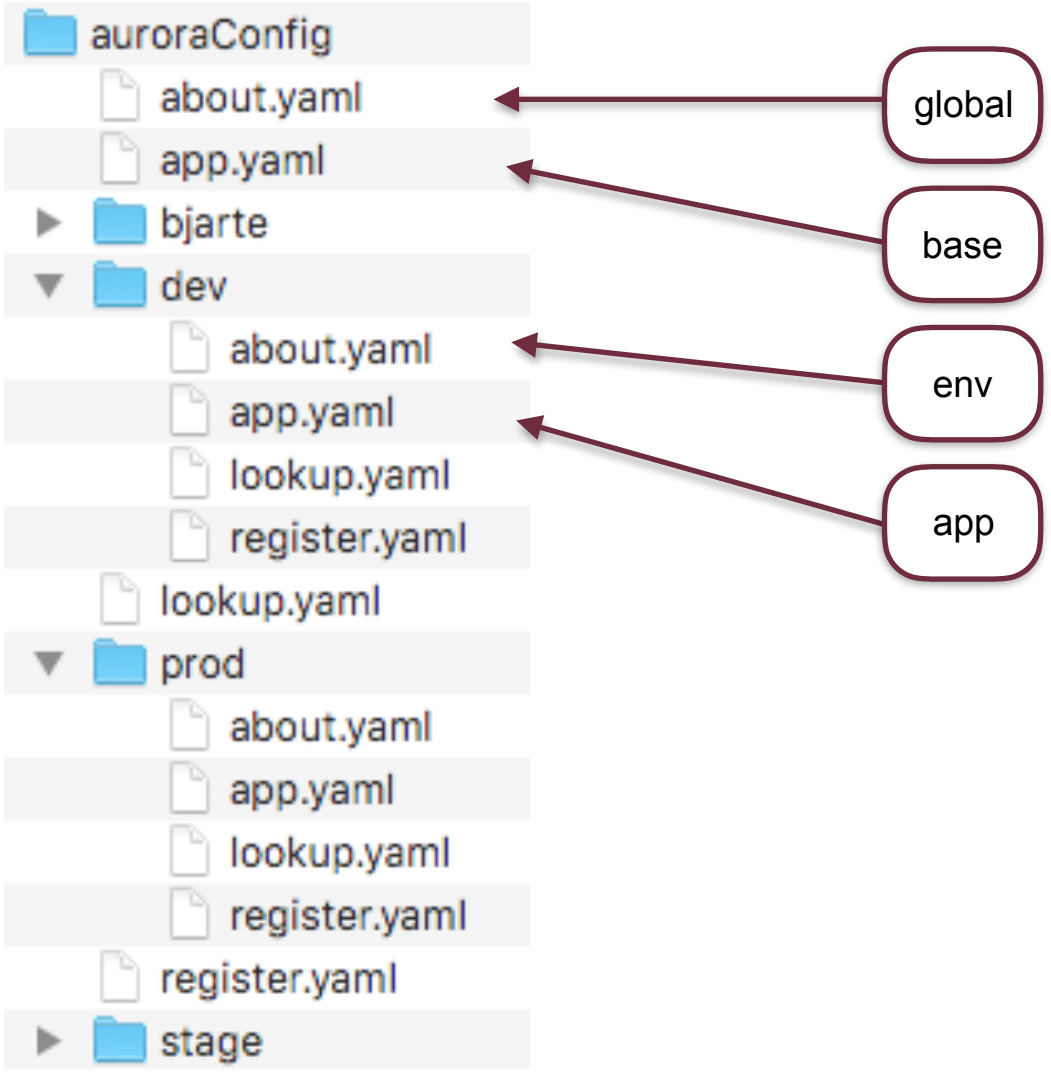

deploying
provisioning


User
2. upgrade
WebDeploy
1. deploy
2.1 deploy
git
application.yaml
1.1. read
DeployTool
1.3. deploy
App
1.2. get DeliveryBundle
1.4.register
DeliveryBundle.zip
application-spec.yaml
lib/app.jar
provides
Nexus
LookupService


User
1. deploy
2. upgrade
Git
ao
AuroraKonsole
App
provides
1.1 deploy
2.1 deploy
1.5 deploy
AuroraConfig
1.2. read
AuroraApi
1.4. add/update objects
kubernetes


ImageStream
version: 1
scheduled import
DockerRegistry
version: 1.2.3
app
version=1.2.3
trigger deploy on new image
contains
DeploymentConfig
rollout app-1
ReplicationController
start
Pod
name: app-1-xxx


ImageStream
version: 1
scheduled import
DockerRegistry
version: 1.3.0
app
version=1.3.0
trigger deploy on new image
contains
DeploymentConfig
rollout app-2
ReplicationController
start
Pod
name: app-2-xxx


auroraConfig
about.yaml
app.yaml
bjarte
dev
about.yaml
app.yaml
lookup.yaml
register.yaml
lookup.yaml
prod
about.yaml
app.yaml
lookup.yaml
register.yaml
register.yaml
stage
global
base
env
app

about.yaml

```
schemaVersion: v1
affiliation: paas
permissions:
  admin: [PAAS_OPS, PAAS_DEV]
splunkIndex: paas-prod
```

dev/about.yaml

```
cluster: dev
splunkIndex: paas-dev
```

app.yaml

```
groupId: no.skatteetaten.kubecon
version: 1
type: deploy
certificate: true
database: true
```

dev/app.yaml

```
version: 1.2
```

```
schemaVersion: v1
affiliation: paas
permissions:
  admin: [PAAS_OPS, PAAS_DEV]
groupId: no.skatteetaten.kubecon
version: 1.2
type: deploy
certificate: true
database: true
cluster: dev
splunkIndex: paas-test
name: app
envName: dev
artifactId: app
```

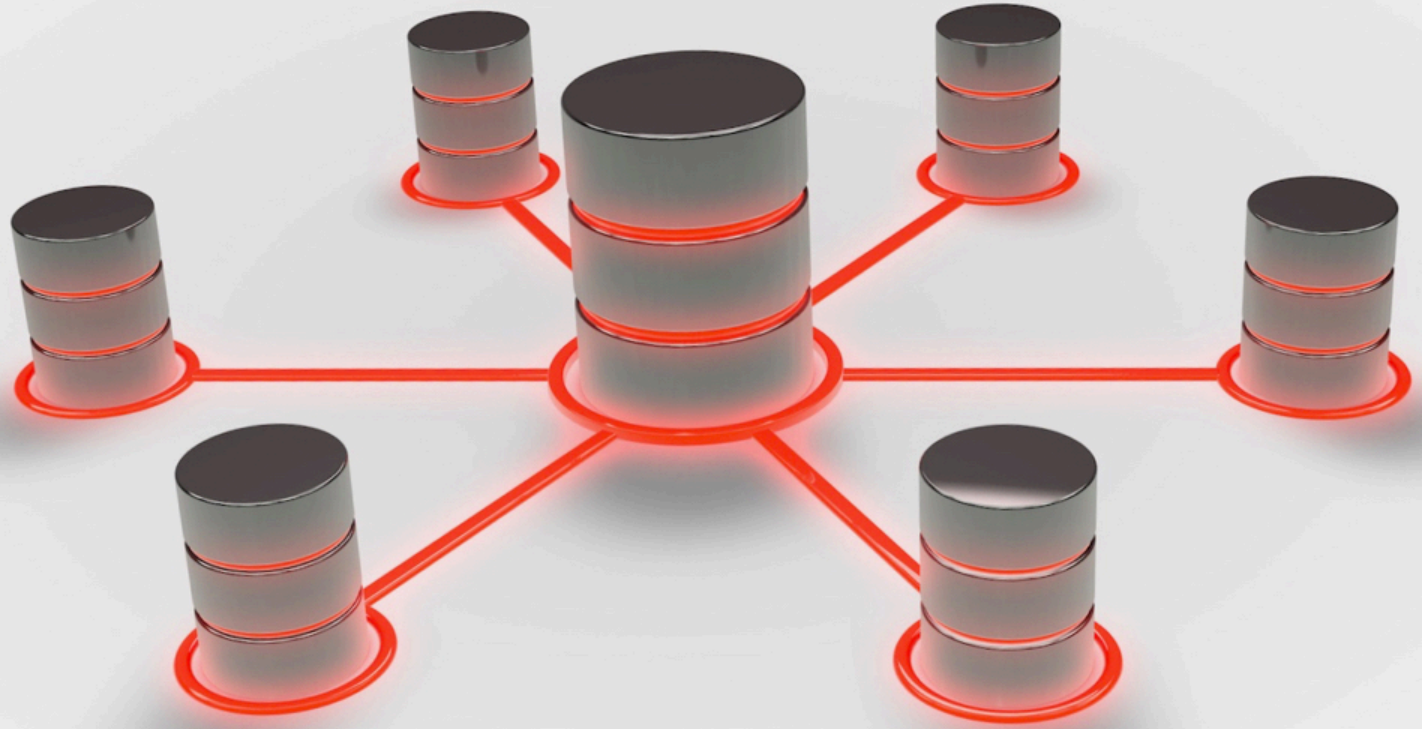

FEEDBACK
ANALYSIS
IDEA
PHONE CALLS
RATING
ADVICE
OPINION
RESPONSE
COMMENT
RESULTS
SURVEY
Q
A

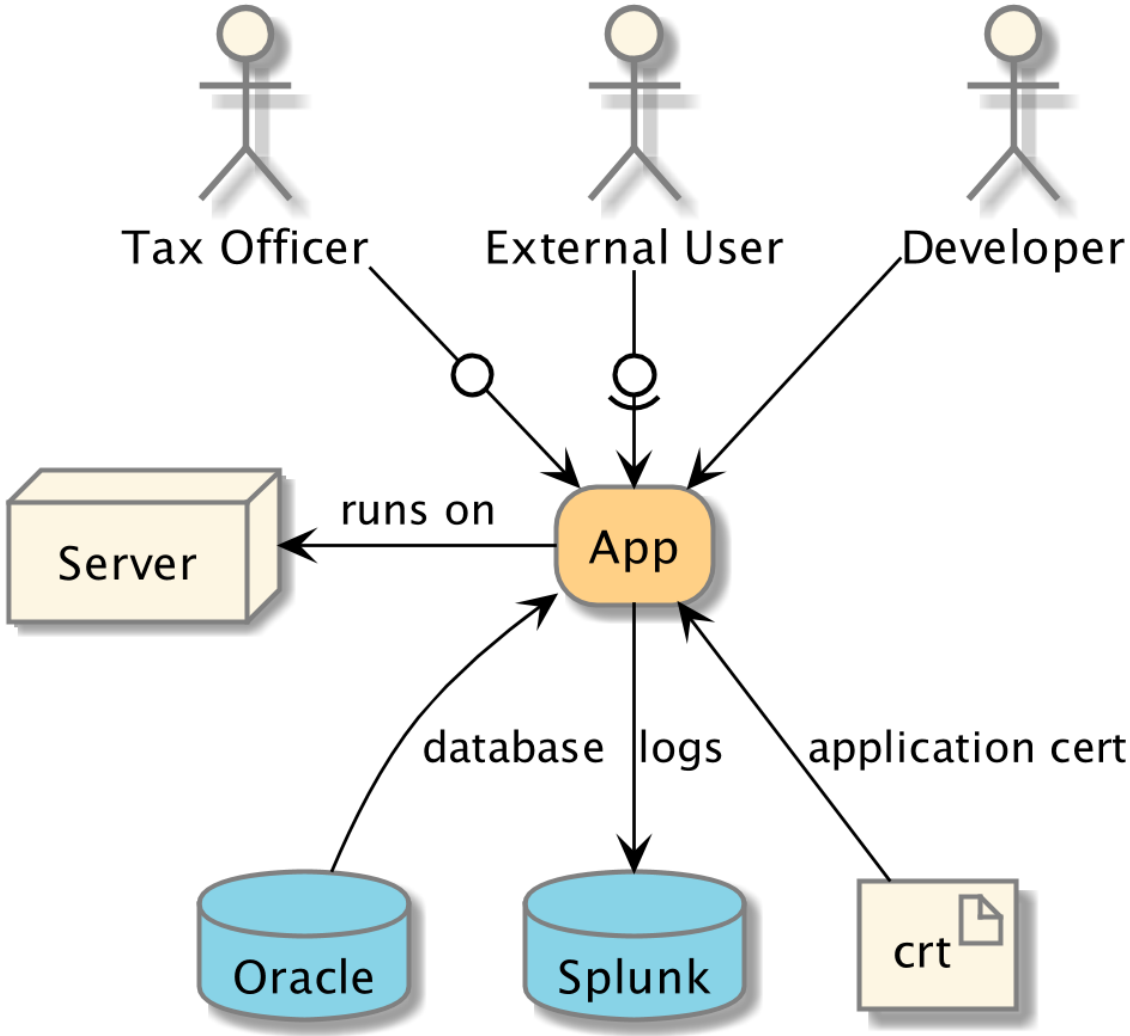
Skatteetaten

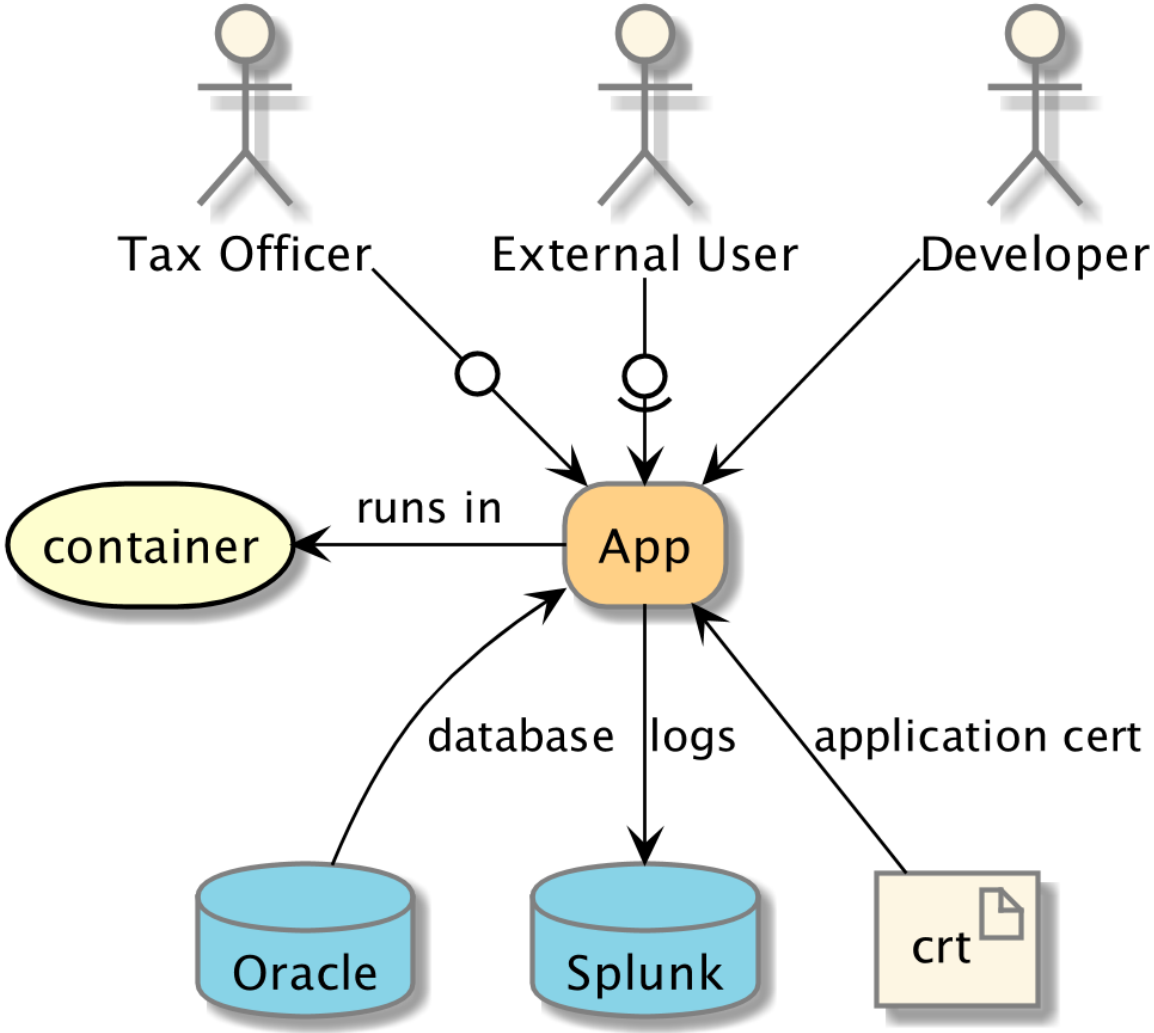
AURORA DEPLOY CONTRACT

- Declarative config in AuroraConfig
- upgrade policy
  - latest, no breaking changes, new features, new bugs, "patches", locked, same as production
- deploy with auroraAPI will overwrite existing state

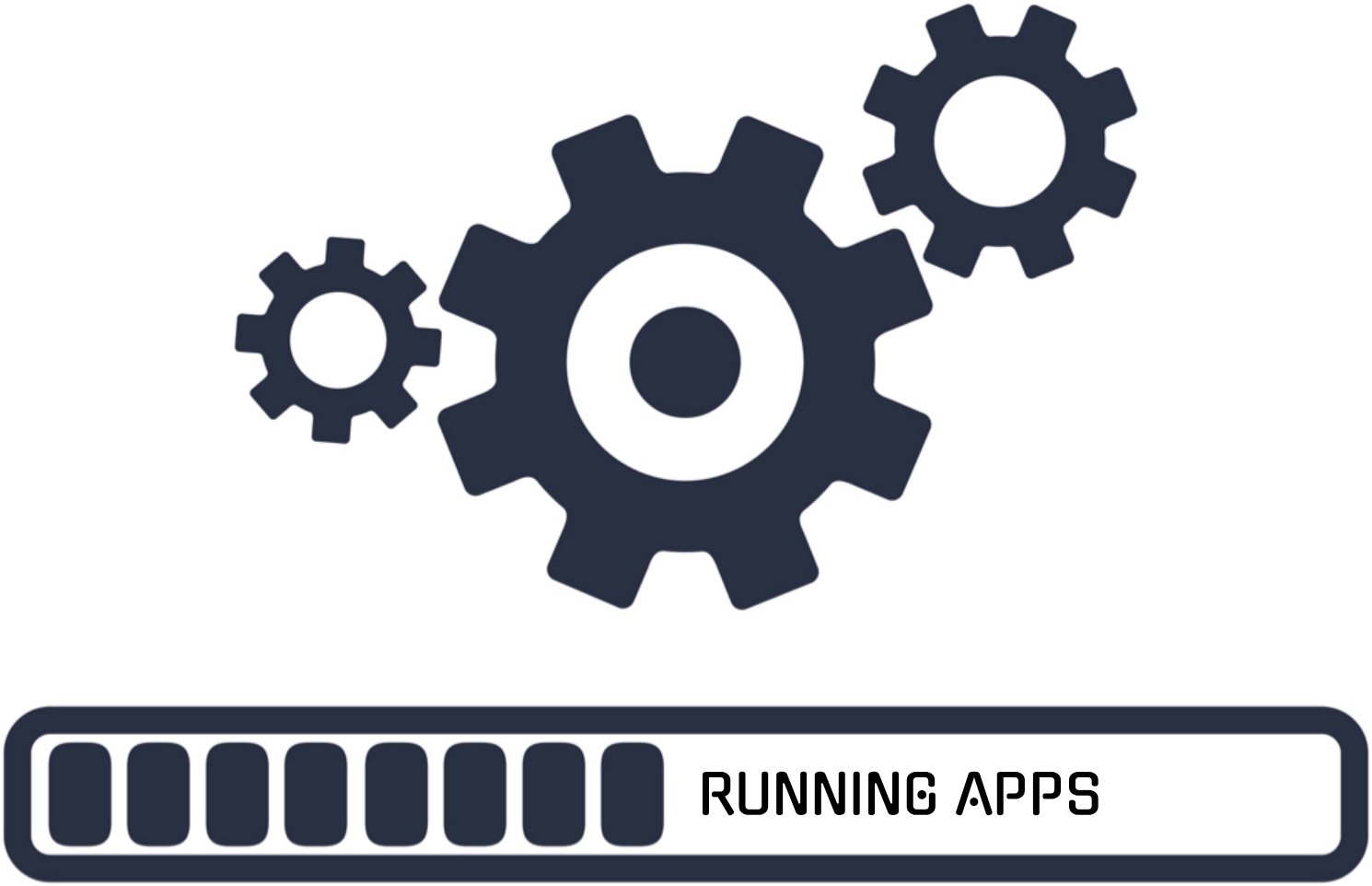
EXTERNAL INTEGRATIONS

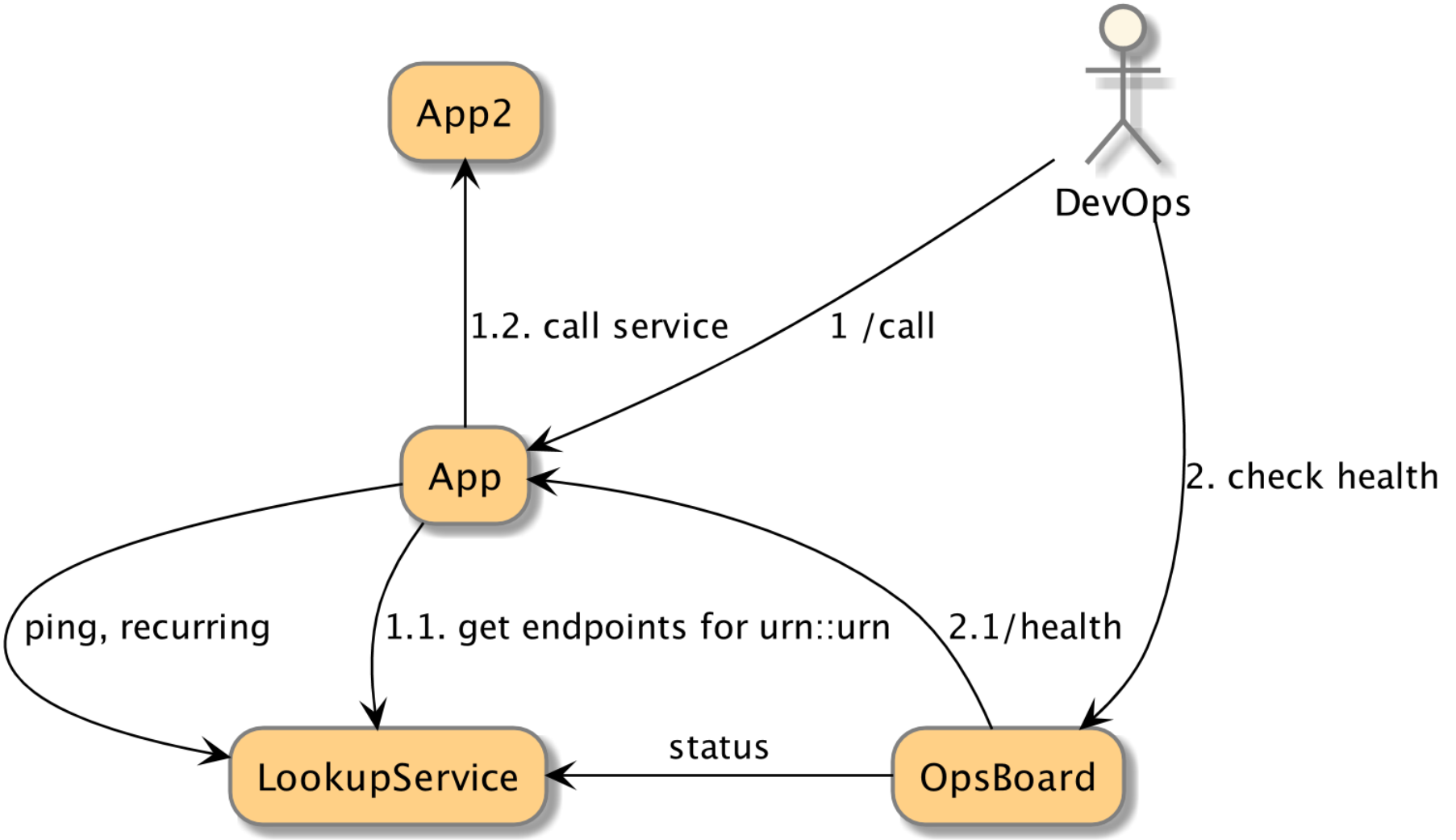


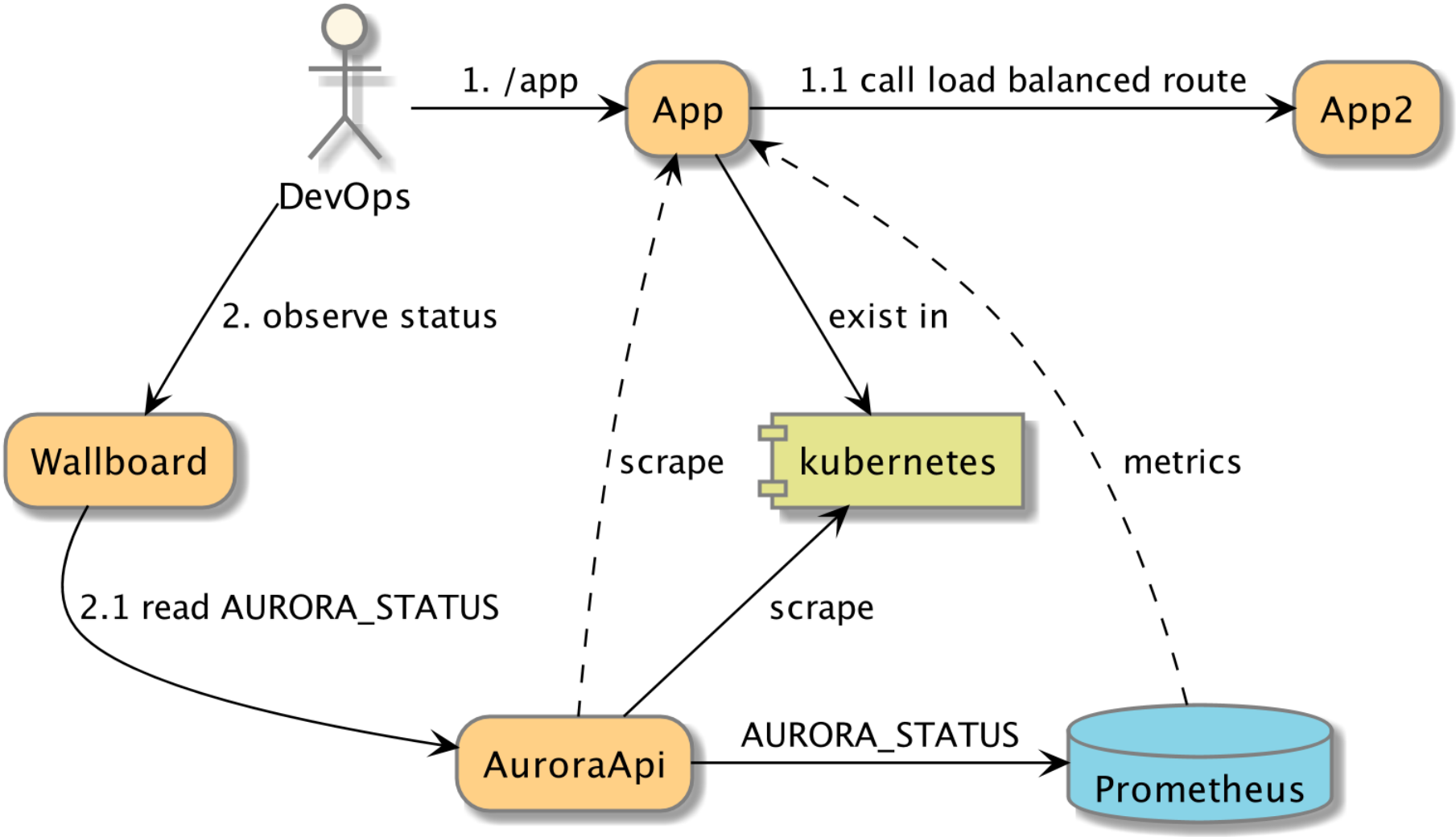

Tax Officer
External User
Developer
Server
runs on
App
database
logs
application cert
Oracle
Splunk
crt

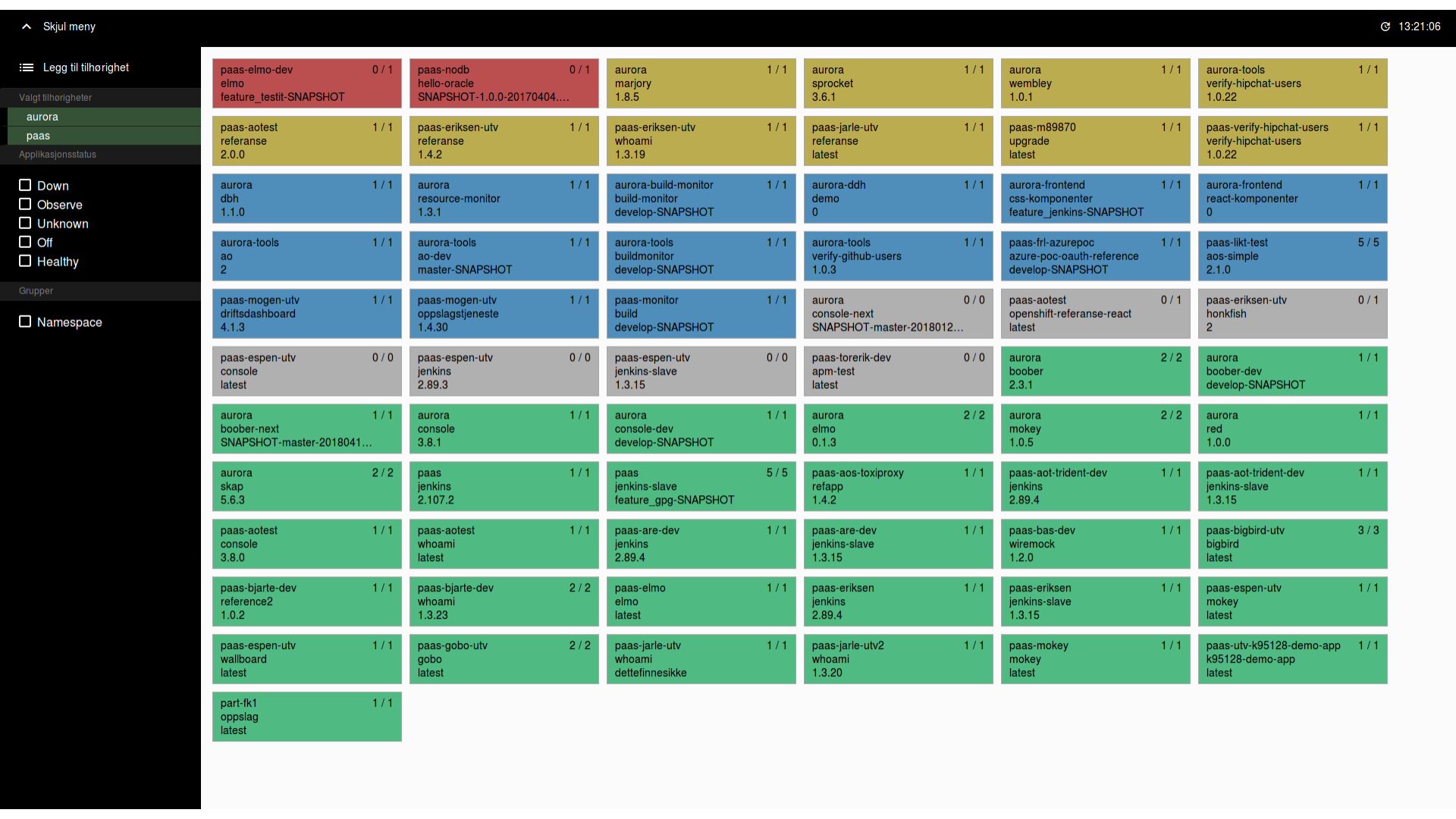

Tax Officer
External User
Developer
container
runs in
App
database
logs
application cert
Oracle
Splunk
crt

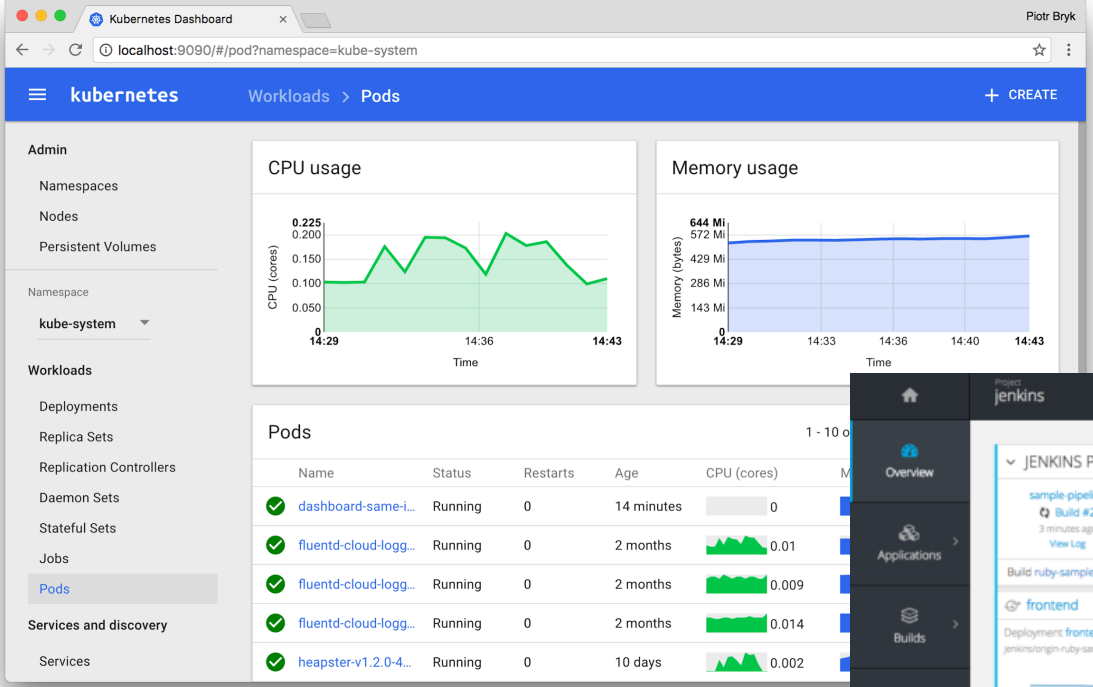

AUTOMATE
STANDARDIZE
11010
10101010

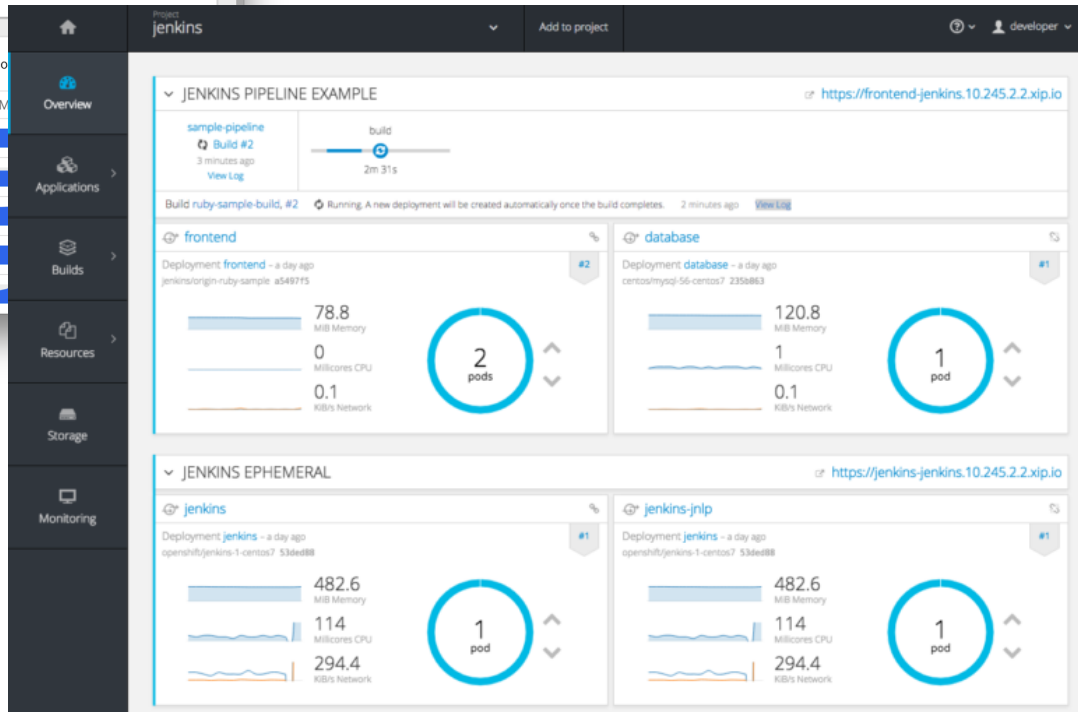

RUNNING APPS


App2
DevOps
1.2. call service
1 /call
App
2. check health
ping, recurring
1.1. get endpoints for urn::urn
2.1/health
LookupService
status
OpsBoard


1. /app
App
1.1 call load balanced route
App2
DevOps
2. observe status
exist in
Wallboard
scrape
kubernetes
metrics
2.1 read AURORA_STATUS
scrape
AURORA_STATUS
AuroraApi
Prometheus


Skjul meny
13:21:06
Legg til tilhørighet
Valgt tilhørigheter
aurora
paas
Applikasjonsstatus
Down
Observe
Unknown
Off
Healthy
Grupper
Namespace
paas-elmo-dev 0 / 1
elmo
feature_testit-SNAPSHOT
paas-nodb 0 / 1
hello-oracle
SNAPSHOT-1.0.0-20170404....
aurora 1 / 1
marjory
1.8.5
aurora 1 / 1
sprocket
3.6.1
aurora 1 / 1
wembley
1.0.1
aurora-tools 1 / 1
verify-hipchat-users
1.0.22
paas-aotest 1 / 1
referanse
2.0.0
paas-eriksen-utv 1 / 1
referanse
1.4.2
paas-eriksen-utv 1 / 1
whoami
1.3.19
paas-jarle-utv 1 / 1
referanse
latest
paas-m89870 1 / 1
upgrade
latest
paas-verify-hipchat-users 1 / 1
verify-hipchat-users
1.0.22
aurora 1 / 1
dbh
1.1.0
aurora 1 / 1
resource-monitor
1.3.1
aurora-build-monitor 1 / 1
build-monitor
develop-SNAPSHOT
aurora-ddh 1 / 1
demo
0
aurora-frontend 1 / 1
css-komponenter
feature_jenkins-SNAPSHOT
aurora-frontend 1 / 1
react-komponenter
0
aurora-tools 1 / 1
ao
2
aurora-tools 1 / 1
ao-dev
master-SNAPSHOT
aurora-tools 1 / 1
buildmonitor
develop-SNAPSHOT
aurora-tools 1 / 1
verify-github-users
1.0.3
paas-frl-azurepoc 1 / 1
azure-poc-oauth-reference
develop-SNAPSHOT
paas-likt-test 5 / 5
aos-simple
2.1.0
paas-mogen-utv 1 / 1
driftsdashboard
4.1.3
paas-mogen-utv 1 / 1
oppslagstjeneste
1.4.30
paas-monitor 1 / 1
build
develop-SNAPSHOT
aurora 0 / 0
console-next
SNAPSHOT-master-2018012...
paas-aotest 0 / 1
openshift-referanse-react
latest
paas-eriksen-utv 0 / 1
honkfish
2
paas-espen-utv 0 / 0
console
latest
paas-espen-utv 0 / 0
jenkins
2.89.3
paas-espen-utv 0 / 0
jenkins-slave
1.3.15
paas-torerik-dev 0 / 0
apm-test
latest
aurora 2 / 2
boober
2.3.1
aurora 1 / 1
boober-dev
develop-SNAPSHOT
aurora 1 / 1
boober-next
SNAPSHOT-master-2018041...
aurora 1 / 1
console
3.8.1
aurora 1 / 1
console-dev
develop-SNAPSHOT
aurora 2 / 2
elmo
0.1.3
aurora 2 / 2
mokey
1.0.5
aurora 1 / 1
red
1.0.0
aurora 2 / 2
skap
5.6.3
paas 1 / 1
jenkins
2.107.2
paas 5 / 5
jenkins-slave
feature_gpg-SNAPSHOT
paas-aos-toxiproxy 1 / 1
refapp
1.4.2
paas-aot-trident-dev 1 / 1
jenkins
2.89.4
paas-aot-trident-dev 1 / 1
jenkins-slave
1.3.15
paas-aotest 1 / 1
console
3.8.0
paas-aotest 1 / 1
whoami
latest
paas-are-dev 1 / 1
jenkins
2.89.4
paas-are-dev 1 / 1
jenkins-slave
1.3.15
paas-bas-dev 1 / 1
wiremock
1.2.0
paas-bigbird-utv 3 / 3
bigbird
latest
paas-bjarte-dev 1 / 1
reference2
1.0.2
paas-bjarte-dev 2 / 2
whoami
1.3.23
paas-elmo 1 / 1
elmo
latest
paas-eriksen 1 / 1
jenkins
2.89.4
paas-eriksen 1 / 1
jenkins-slave
1.3.15
paas-espen-utv 1 / 1
mokey
latest
paas-espen-utv 1 / 1
wallboard
latest
paas-gobo-utv 2 / 2
gobo
latest
paas-jarle-utv 1 / 1
whoami
dettefinnesikke
paas-jarle-utv2 1 / 1
whoami
1.3.20
paas-mokey 1 / 1
mokey
latest
paas-utv-k95128-demo-app 1 / 1
k95128-demo-app
latest
part-fk1 1 / 1
oppslag
latest

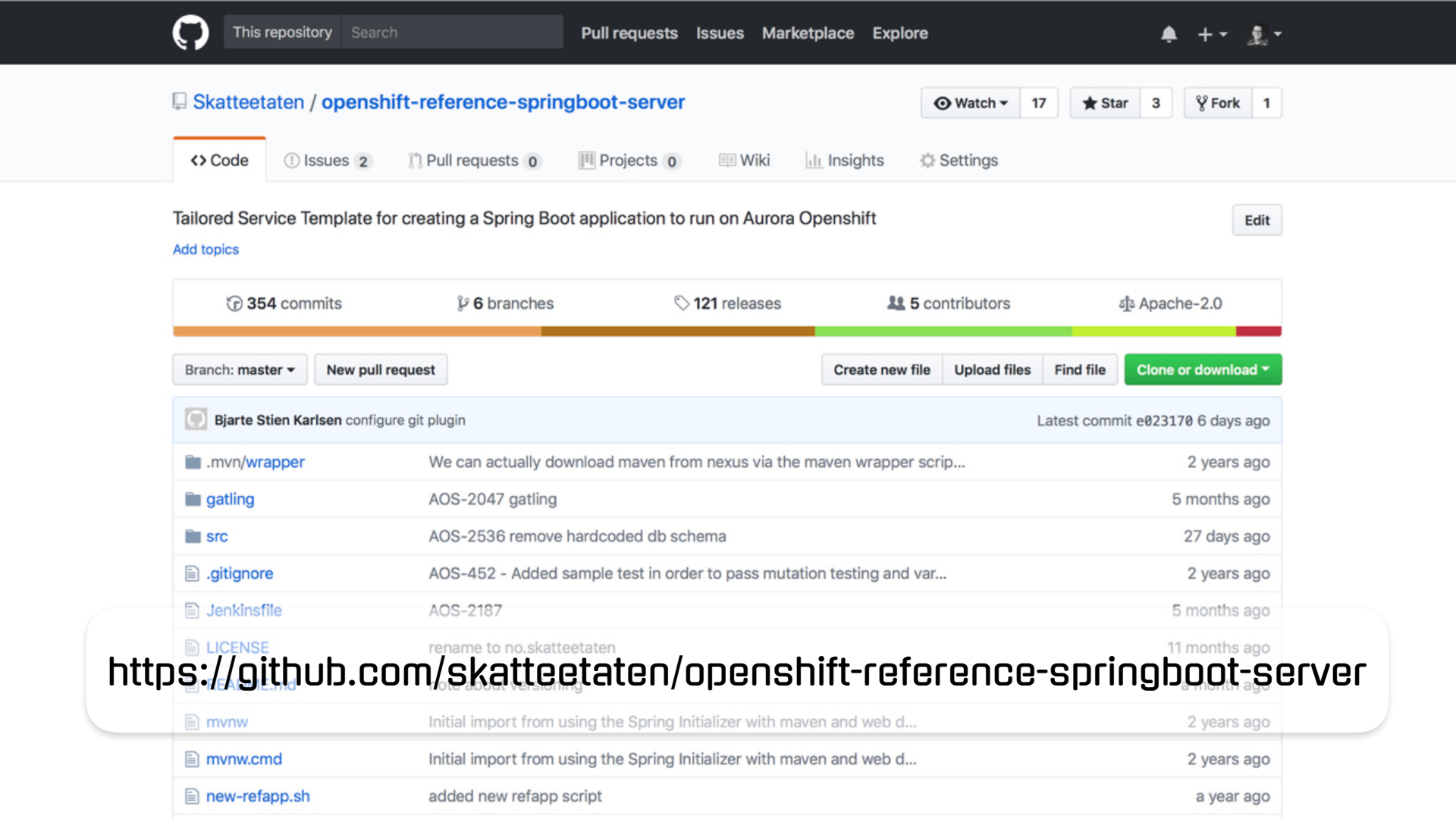

Kubernetes Dashboard
Piotr Bryk
localhost:9090/#/pod?namespace=kube-system
kubernetes
Workloads > Pods
CREATE
Admin
Namespaces
Nodes
Persistent Volumes
Namespace
kube-system
Workloads
Deployments
Replica Sets
Replication Controllers
Daemon Sets
Stateful Sets
Jobs
Pods
Services and discovery
Services
CPU usage
CPU (cores)
0.225
0.200
0.150
0.100
0.050
0
14:29
14:36
14:43
Time
Memory usage
Memory (bytes)
644 Mi
572 Mi
429 Mi
286 Mi
143 Mi
0
14:29
14:33
14:36
14:40
14:43
Time
Pods
1 - 10 o
Name
Status
Restarts
Age
CPU (cores)
dashboard-same-i...
Running
0
14 minutes
0
fluentd-cloud-logg...
Running
0
2 months
0.01
fluentd-cloud-logg...
Running
0
2 months
0.009
fluentd-cloud-logg...
Running
0
2 months
0.014
heapster-v1.2.0-4...
Running
0
10 days
0.002


Project
jenkins
Add to project
developer
Overview
Applications
Builds
Resources
Storage
Monitoring
JENKINS PIPELINE EXAMPLE
https://frontend-jenkins.10.245.2.2.xip.io
sample-pipeline
Build #2
3 minutes ago
View Log
build
2m 31s
Build ruby-sample-build, #2
Running. A new deployment will be created automatically once the build completes.
2 minutes ago
View Log
frontend
Deployment frontend – a day ago
jenkins/origin-ruby-sample a5497f5
#2
78.8
MiB Memory
0
Millicores CPU
0.1
KiB/s Network
2
pods
database
Deployment database – a day ago
centos/mysql-56-centos7 235b863
#1
120.8
MiB Memory
1
Millicores CPU
0.1
KiB/s Network
1
pod
JENKINS EPHEMERAL
https://jenkins-jenkins.10.245.2.2.xip.io
jenkins
Deployment jenkins – a day ago
openshift/jenkins-1-centos7 53ded88
#1
482.6
MiB Memory
114
Millicores CPU
294.4
KiB/s Network
1
pod
jenkins-jnlp
Deployment jenkins – a day ago
openshift/jenkins-1-centos7 53ded88
#1
482.6
MiB Memory
114
Millicores CPU
294.4
KiB/s Network
1
pod

Skatteetaten

AURORA RUNTIME CONTRACT

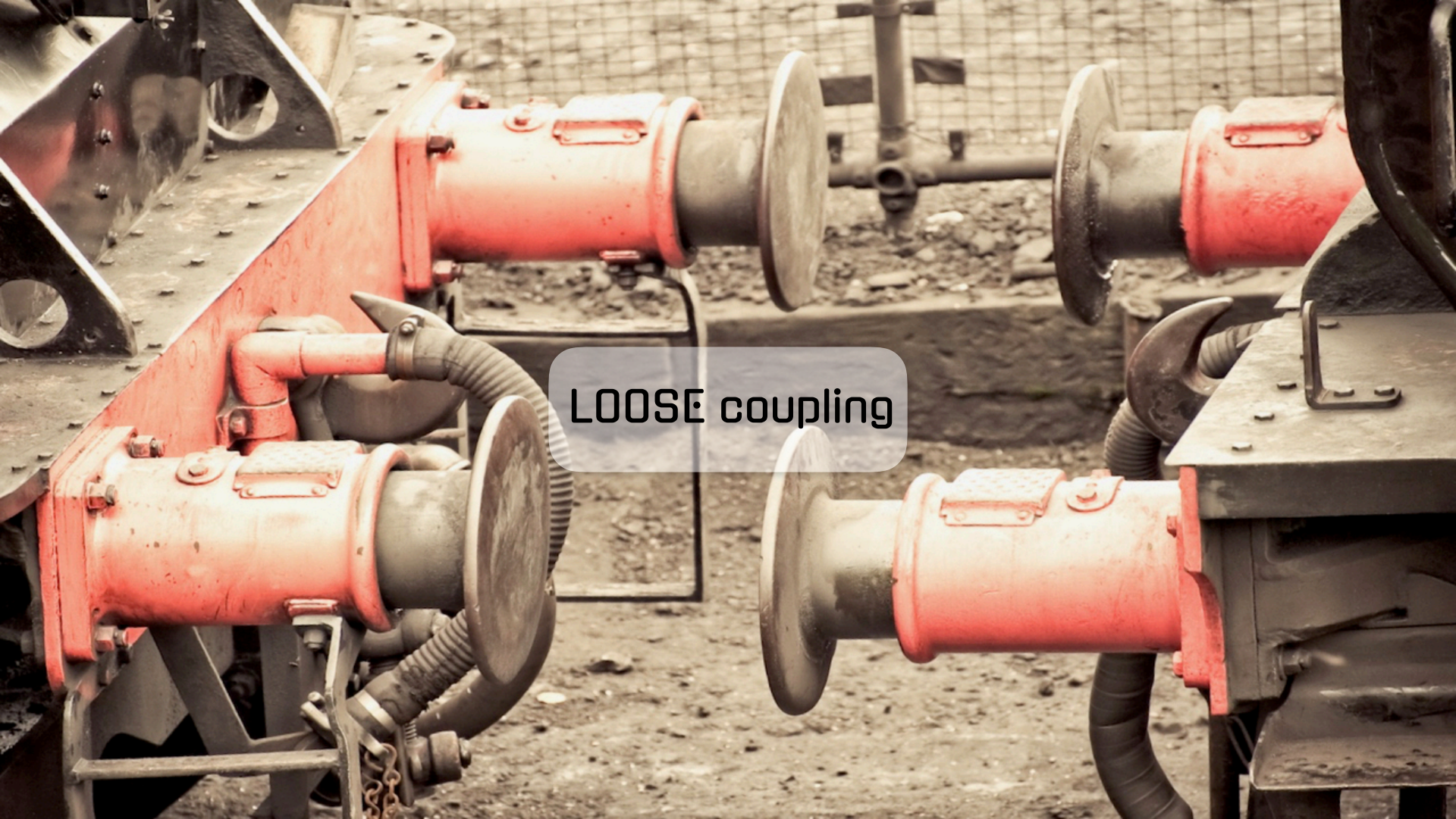
- microservice
- 12-factor (best practice cloud-ready)
  - will be killed on node evacuation
- separate app/management traffic
- managementInterface
  - /actuator
  - /info
  - /health
  - /prometheus
  - /env
- use provided $LOGBACK_FILE

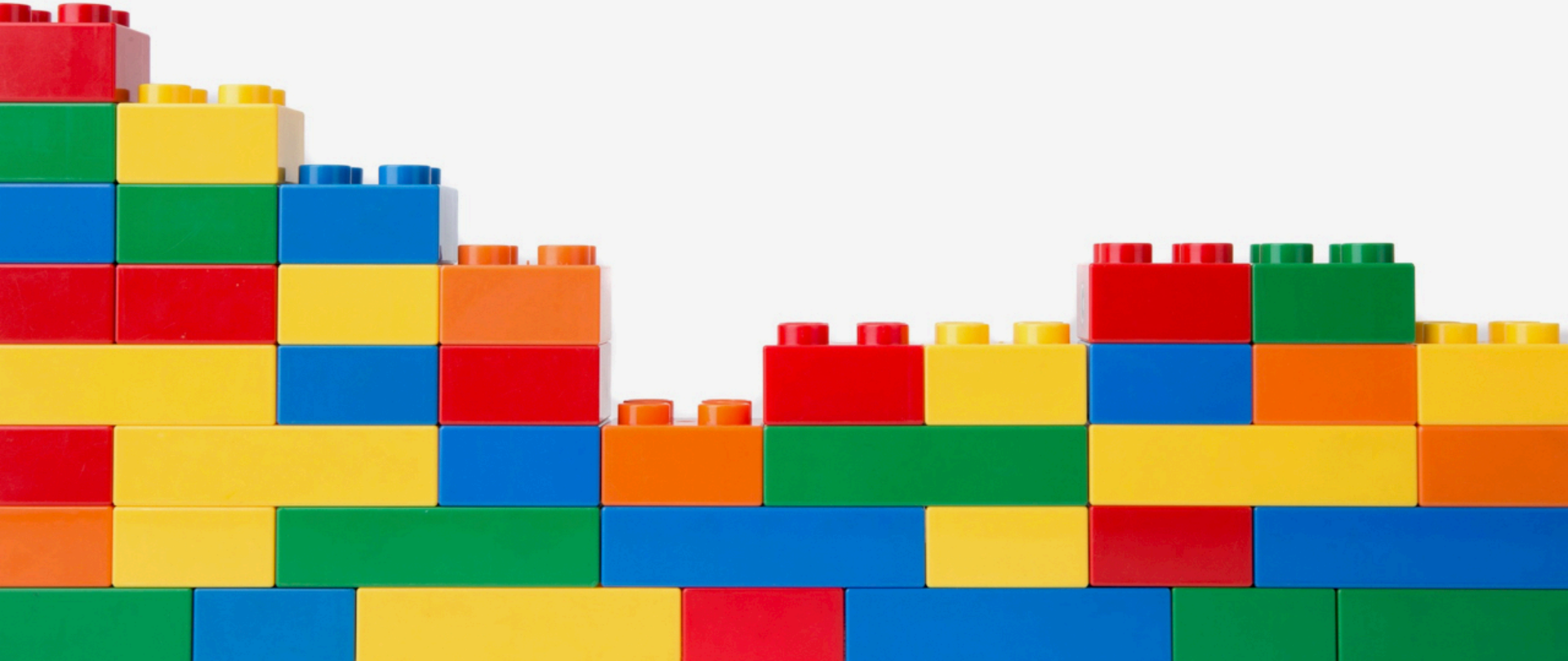


https://github.com/skatteetaten/openshift-reference-springboot-server


delete
enter
return
50
GENERAL LESSONS

CONTRACTS


LOOSE coupling




KUBERNEEEEEEEEEE..

THE AURORA PLATFORM

Faster development and more efficient ops

- PaaS built upon OpenShift
- At the core of the plattform is the declerative config format AuroraConfig and the AuroraAPI.
- The AuroraAPI supports deploying applications and observing their status while running.
- Building applications from source is done in the AuroraPipeline that uses a central build logic.

Why did you make things inhouse?

- Both Kubernetes and OpenShift lack a concept we have called **affiliation**. That is groups of people that can administer or view objects for several projects. We have several different development teams that work on our clusters and we want them to be able to work in **isolation**.
- The ability to deploy applications to **several clusters** in one command is highly desired within our organization. Our network infrastructure implies that we need to have multiple clusters.
- [illegible] projects we want to avoid duplication. Our **declarative** config format AuroraConfig supports **composition** with **sane defaults**

https://skatteetaten.github.io/aurora

In order to avoid 'wall-of-yaml' we use a declarative, composable configuration format with sane defaults.

Bjarte Karlsen, Technical Architect NTA

How do we deploy

An deploy starts with triggering the AuroraAPI from either of the userfacing clients AO, AuroraKonsole or AuroraPipeline. The API will then extract and merge AuroraConfig in order to create a AuroraDeploymentSpec.

Synchrous integrations are run and the result of both are assembled into Kubernetes objects that are applied to the cluster. Async integrations will that use the