"Zero to Hero in Minutes"

Chakradhar Rao Jonagam

Solutions Architect & CNCF Ambassador

@debianmaster

Want to use k8s?

+

+

=

Ready for primetime

What is openshift?

OPENSHIFT origin =  +  + 

k8s

dev-tools

awesomeness

Analogy

= Kernel for you DC

OPENSHIFT origin = O/S for your DC

What does openshift add ?

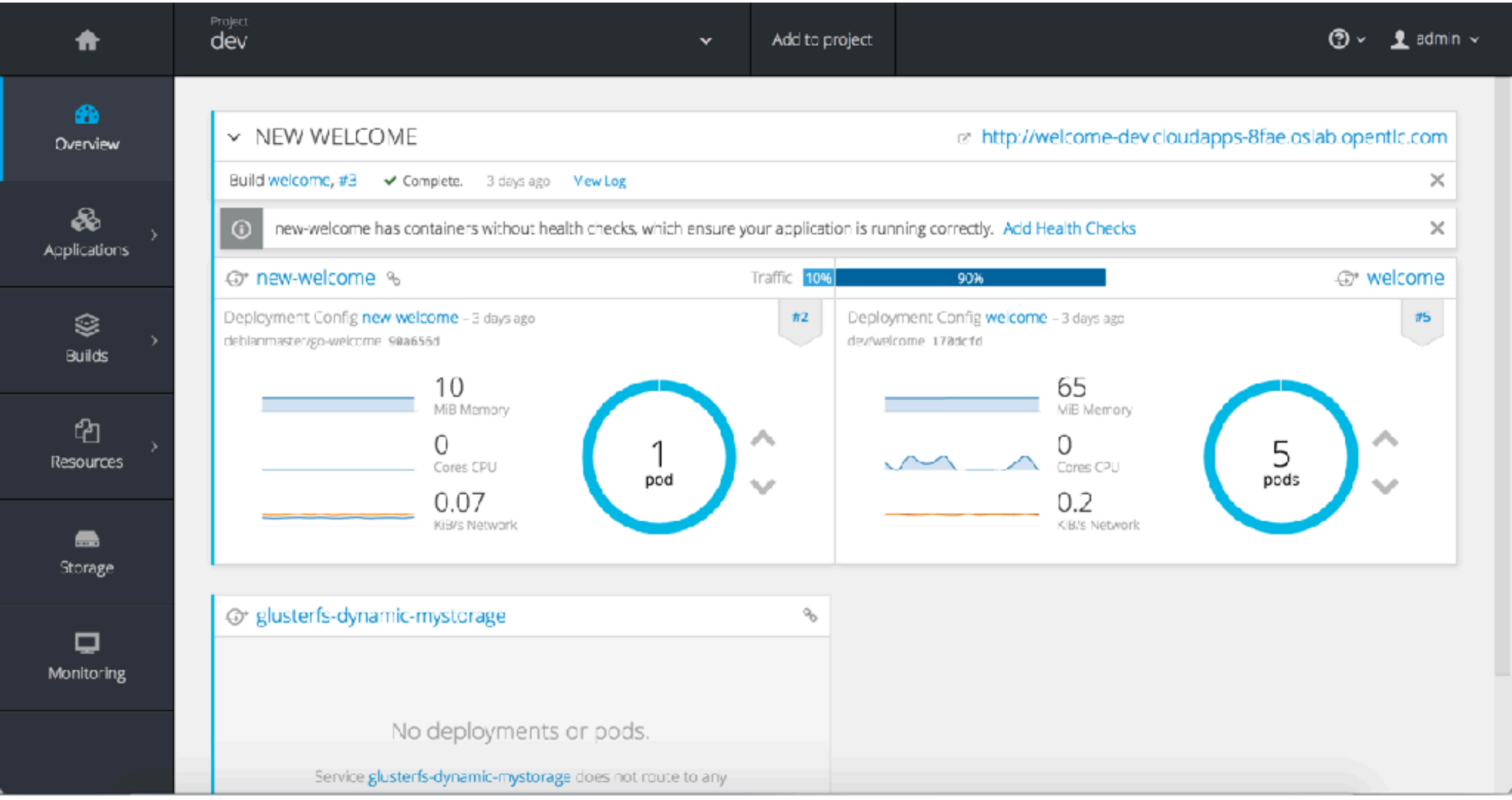
-> Developer tools
-> ci/cd work workflow
-> Advanced Multi-tenancy
-> Advanced User Interface
-> Registry & trusted base images
-> Many more .....

Start a cluster

```
$ oc cluster up
```

oc + docker

UI for k8s


Project
dev
Add to project
admin
Overview
Applications
Builds
Resources
Storage
Monitoring
NEW WELCOME
http://welcome-dev.cloudapps-8fae.oslab.opentlc.com
Build welcome, #3
Complete.
3 days ago
View Log
new-welcome has containers without health checks, which ensure your application is running correctly.
Add Health Checks
new-welcome
Traffic
10%
90%
welcome
Deployment Config new welcome – 3 days ago
#2
10
MiB Memory
0
Cores CPU
0.07
KiB/s Network
1
pod
Deployment Config welcome – 3 days ago
#5
65
MiB Memory
0
Cores CPU
0.2
KiB/s Network
5
pods
glusterfs-dynamic-mystorage
No deployments or pods.
Service glusterfs-dynamic-mystorage does not route to any

Start from anywhere

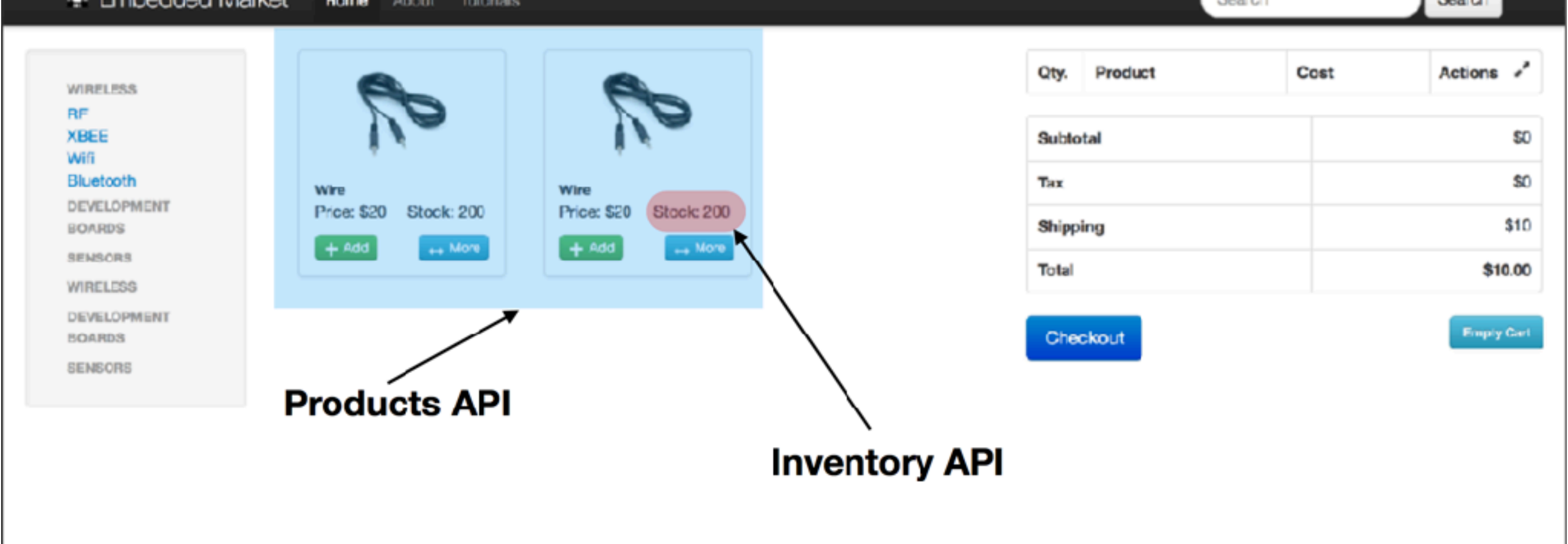
Docker image

Source code

Dockerfile

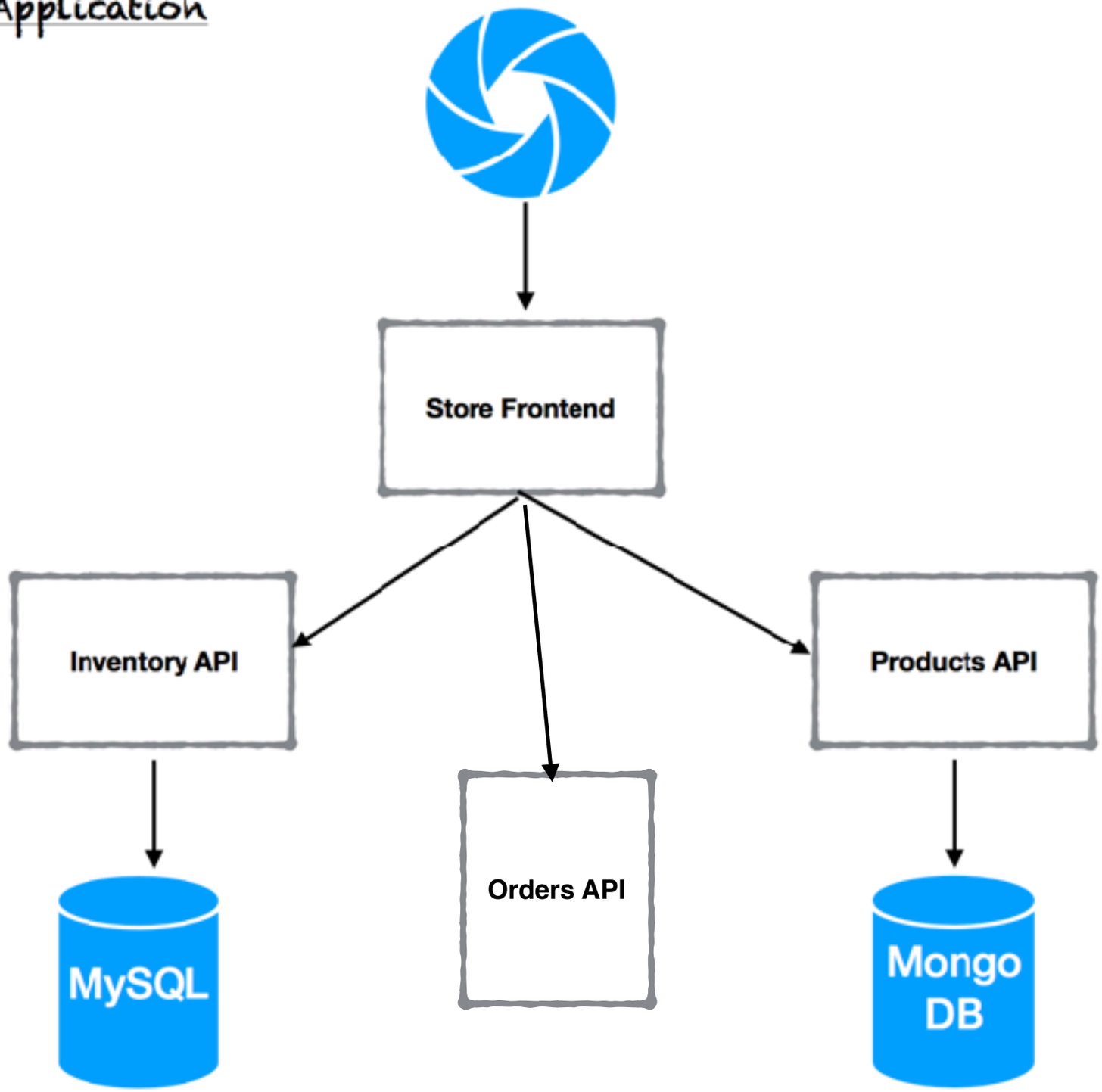
JAR

WAR

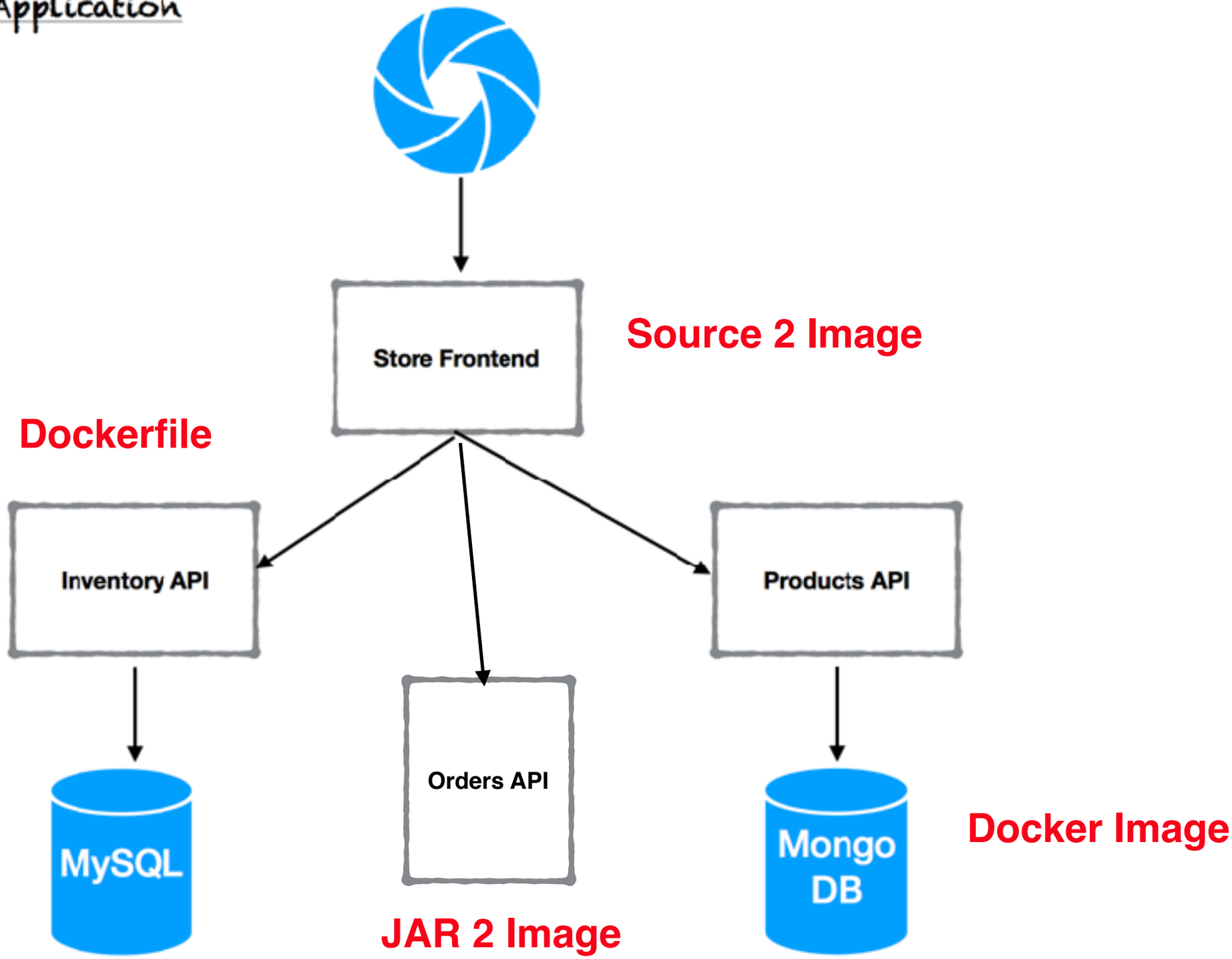


UI microservice

Example Application



Example Application



From Docker Image

```
$ oc new-app docker.io/mongodb
```

From Dockerfile

$ oc new-app https://github.com/i63/store-inventory —strategy=docker

Build via Dockerfile

Follow

```
Cloning "https://github.com/i63/store-inventory" ...
        Commit: 290576d79858aff432a6cd3668ffa4fc426fe10c (Merge branch 'master' of https://github.com/i63/store-inventory)
        Author: tetradev1 <9chakri+ds@gmail.com>
        Date:   Sat Jun 24 15:46:32 2017 -0700
Pulling image golang@sha256:9c4e698d6d1378cea1a7dbdbce2688b50f458d3a09a9f9cd8e1825c187dc3c43 ...
Pulled 0/7 layers, 1% complete
Pulled 1/7 layers, 27% complete
Pulled 2/7 layers, 41% complete
Pulled 3/7 layers, 43% complete
Pulled 4/7 layers, 57% complete
Pulled 5/7 layers, 74% complete
Pulled 6/7 layers, 95% complete
Pulled 7/7 layers, 100% complete
Extracting
Step 1 : FROM golang@sha256:9c4e698d6d1378cea1a7dbdbce2688b50f458d3a09a9f9cd8e1825c187dc3c43
 ---> 6ce094895555
Step 2 : RUN mkdir /app
 ---> Running in 10c9e1c3a518
 ---> d8fdd3b1ba08
Removing intermediate container 10c9e1c3a518
Step 3 : RUN go get github.com/gin-gonic/gin && go get github.com/go-sql-driver/mysql
 ---> Running in aa2e6323df48
```

Source 2 image

```
$ oc new-app \
  https://github.com/i63/store-products \
  --strategy=source
```

Source 2 image process

Follow

```
Cloning "https://github.com/i63/store-products" ...
        Commit: 8d7761e1e8905600f2c7ebebba588a337d065060 (Update README.md)
        Author: Chakradhar Rao Jonagam <debianmaster@users.noreply.github.com>
        Date:   Sat Jun 24 15:45:23 2017 -0700
---> Installing application source ...
---> Building your Node application from source
mongodb@2.2.30 node_modules/mongodb
├── es6-promise@3.2.1
├── readable-stream@2.2.7 (inherits@2.0.3, buffer-shims@1.0.0, process-nextick-args@1.0.7, util-deprecate@1.0.2, core-util-is@1.0.2, isarray@1.0.0, string_decoder@1.0.3)
└── mongodb-core@2.1.14 (require_optional@1.0.1, bson@1.0.4)
express@3.21.2 node_modules/express
├── escape-html@1.0.2
├── merge-descriptors@1.0.0
├── cookie@0.1.3
├── commander@2.6.0
├── cookie-signature@1.0.6
├── proxy-addr@1.0.10 (forwarded@0.1.0, ipaddr.js@1.0.5)
├── send@0.13.0 (destroy@1.0.3, statuses@1.2.1, ms@0.7.1, mime@1.3.4, on-finished@2.3.0, http-errors@1.3.1)
├── mkdirp@0.5.1 (minimist@0.0.8)
└── connect@2.30.2 (bytes@2.1.0, pause@0.1.0, on-headers@1.0.1, vhost@3.0.2, basic-auth-connect@1.0.0, cookie-parser@1.3.5, qs@4.0.0, connect-timeout@1.6.2, serve-favicon@2.3.2, http-errors@1.3.1, response-time@2.3.2, morgan@1.6.1, finalhandler@0.4.0, express-session@1.11.3, type-is@1.6.15, method-override@2.3.9, serve-static@1.10.3, multiparty@3.3.2, compression@1.5.2, errorhandler@1.4.3, csurf@1.8.3, body-parser@1.13.3, serve-index@1.7.3)
Pushing image 172.30.142.160:5000/dev/store-products:latest ...
Pushed 0/5 layers, 3% complete
Pushed 1/5 layers, 23% complete
Pushed 2/5 layers, 49% complete
Pushed 3/5 layers, 70% complete
Pushed 4/5 layers, 96% complete
Pushed 5/5 layers, 100% complete
Push successful
```

From WAR/JAR file

```
$ oc new-build \
    --image-stream=tomat \
    --binary=true
    --name=myapp

$ oc start-build myapp --from-dir=.
```

CI/CD workflow setup

```
jenkinsfile: |-

        node('') {
          stage 'buildInDevelopment'

          stage 'deployInDevelopment'

          stage 'deployInQA'

      }
```

```
$ oc create -f pipeline.yaml
```

CI/CD workflow setup

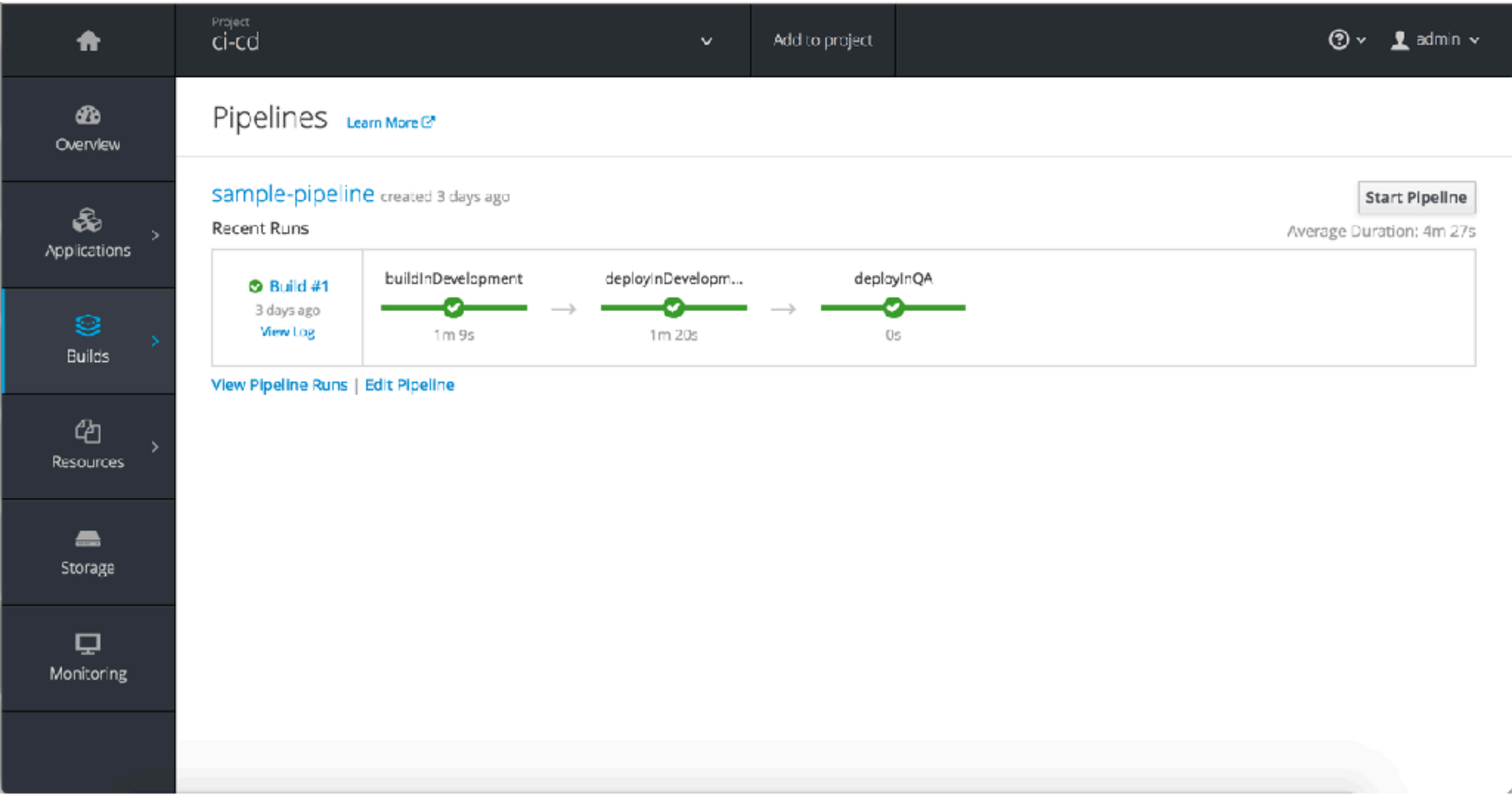
```
jenkinsfile: |-
        node('') {

          stage 'buildInDevelopment'
            openshiftBuild(namespace:'dev',buildConfig: ‘store-frontend', showBuildLogs: ‘true')


          stage 'deployInDevelopment'
            openshiftDeploy(namespace:'dev',deploymentConfig: 'store-frontend')
            openshiftScale(namespace:'dev',deploymentConfig: 'store-frontend',replicaCount: ‘5')


        stage 'deployInQA'
          input ‘Proceed ?'
          openshiftTag(namespace: 'dev', sourceStream: 'store-frontend',
               sourceTag: 'latest', destinationStream: 'store-frontend', destinationTag: 'promoteToQA')
          openshiftDeploy(namespace:'qa',deploymentConfig: 'store-frontend')
}
```

Jenkins Integration



Summary

With Openshift Origin you can use k8s without having to spend time building tooling.

Hence.. Zero to Hero in Minutes

“Thank you!”

@debianmaster