Continuous Delivery Pipeline in Mixed Environments

Presented By:
Sergey Gerasimov

Agenda

1. **Continuous Delivery** and **its cost**
2. **Docker** to the rescue and its mechanics
3. **Rancher** for environment orchestration
4. **Ansible** to reproduce infrastructure
5. **Demo** of their combination

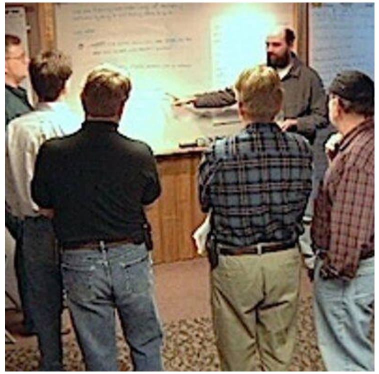
Agile Manifesto

Our highest priority is to satisfy the customer through **early** and **continuous delivery** of **valuable software**

we choose

Continuous Delivery

process



non-Continuous Delivery

Delivery in the end of iteration?

- late feedback
- defects found to late
- last minute fixes
- high risk of change
- code freeze
- night deployment
- whole team meeting

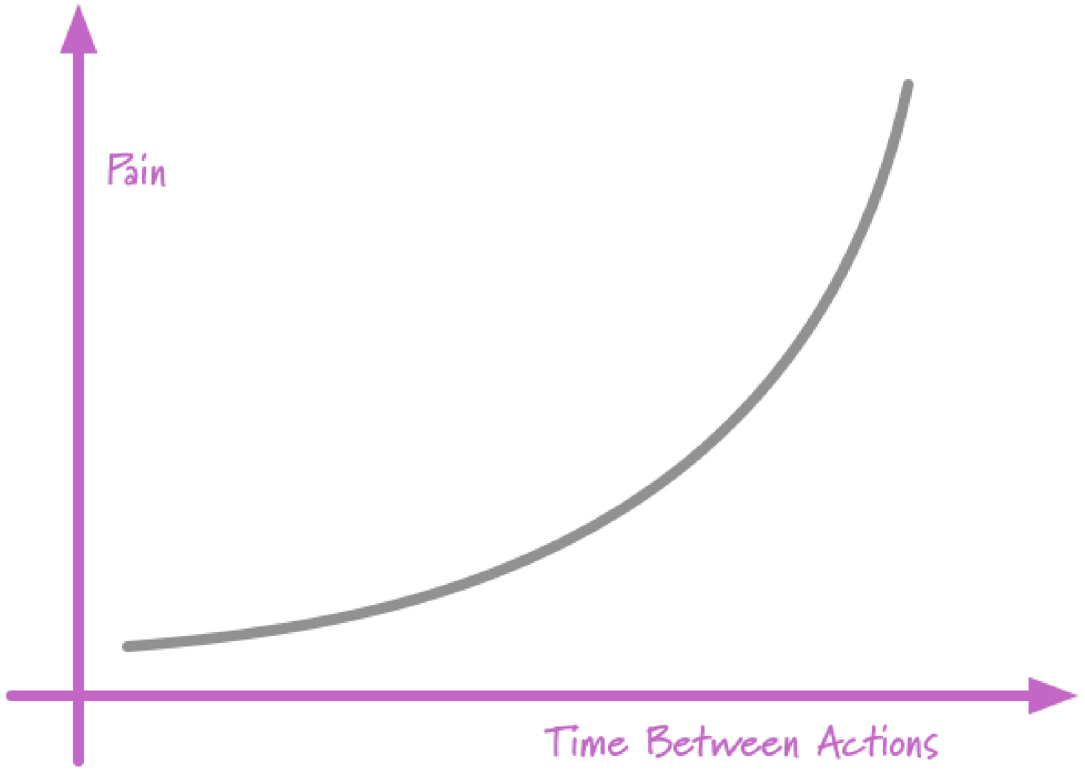
Your feeling:

➡ **Time & Money loss!**

Continuous Delivery

If it hurts,

do it more often



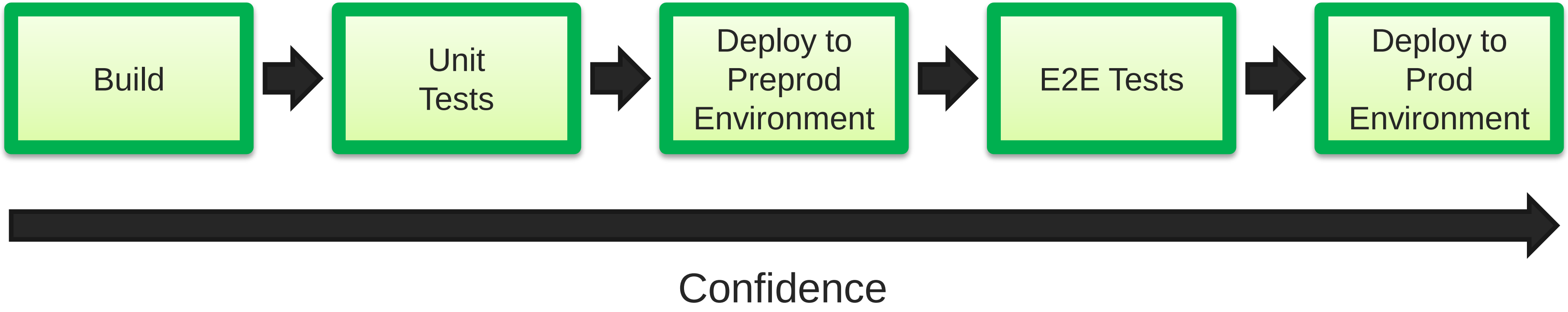
Continuous Delivery

Goal:

Be able to **deliver product** at least **once a day**

Continuous Delivery Pipeline

1. Describe how feature moves from "idea" to "value"
2. Automate it!



Continuous Delivery

Requirements?

1. Continuous Integration
2. Automatic testing
3. Enough test coverage
4. Team's expertise

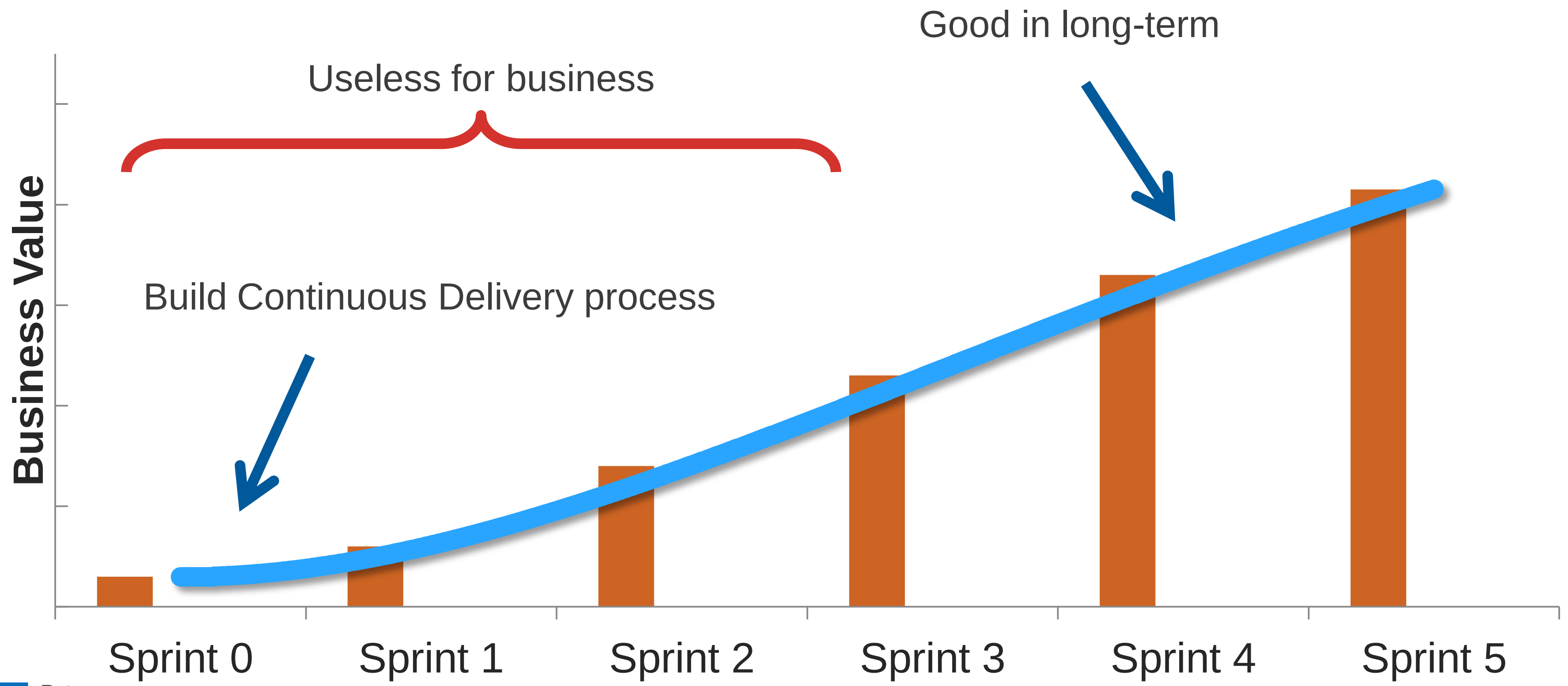
5. Easy to provision **production like** environments
6. Simple deployment
7. Fast deployment
8. Fault-tolerant infrastructure

Our focus

Return on Intelligence

Early delivery

Sprint 0 - investment into future



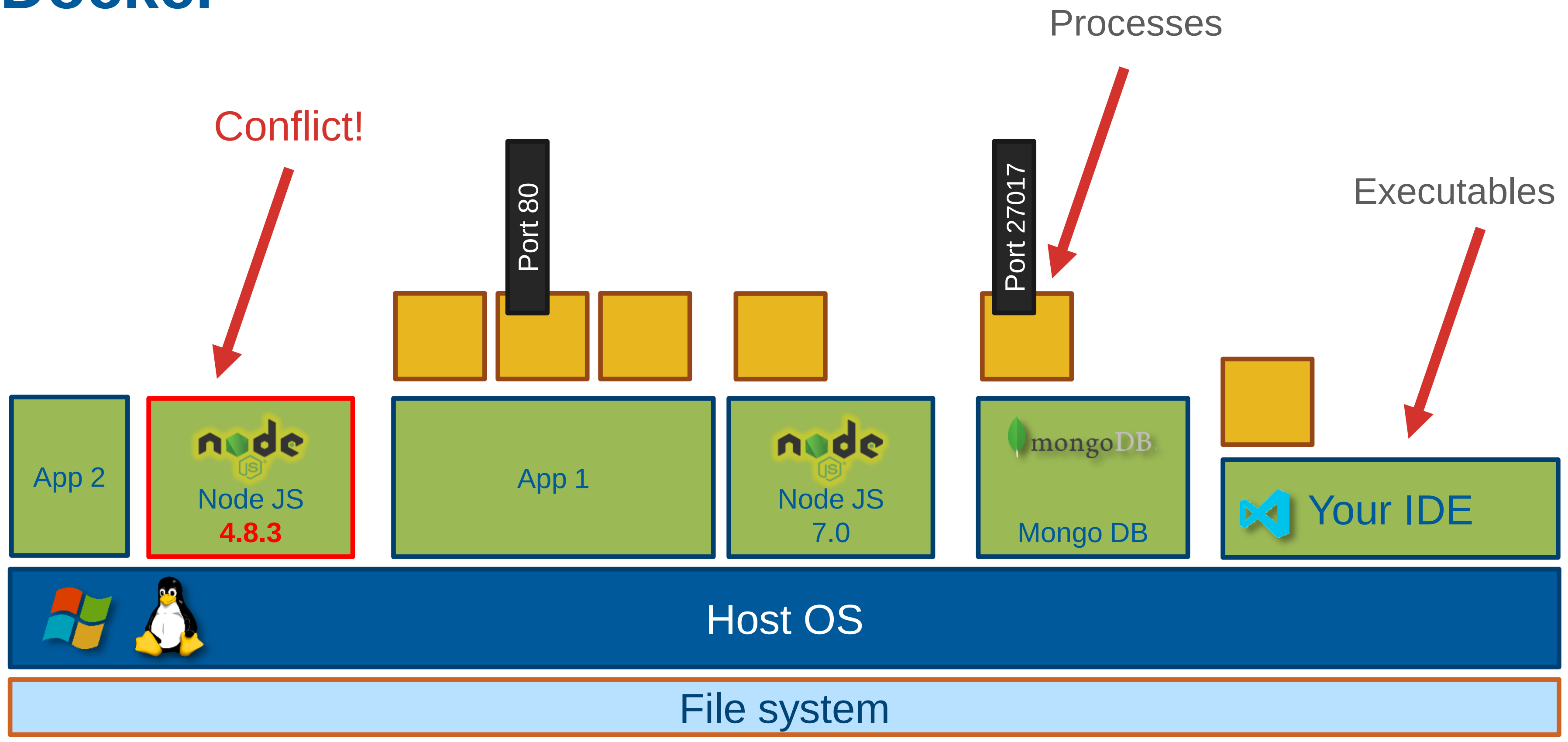
What are our goals?

1. Provision Test and Production environments (**Continuous Delivery**)
2. Provision Continuous Delivery infrastructure (**Early Delivery**)
3. Continuous Delivery pipeline for a sample application

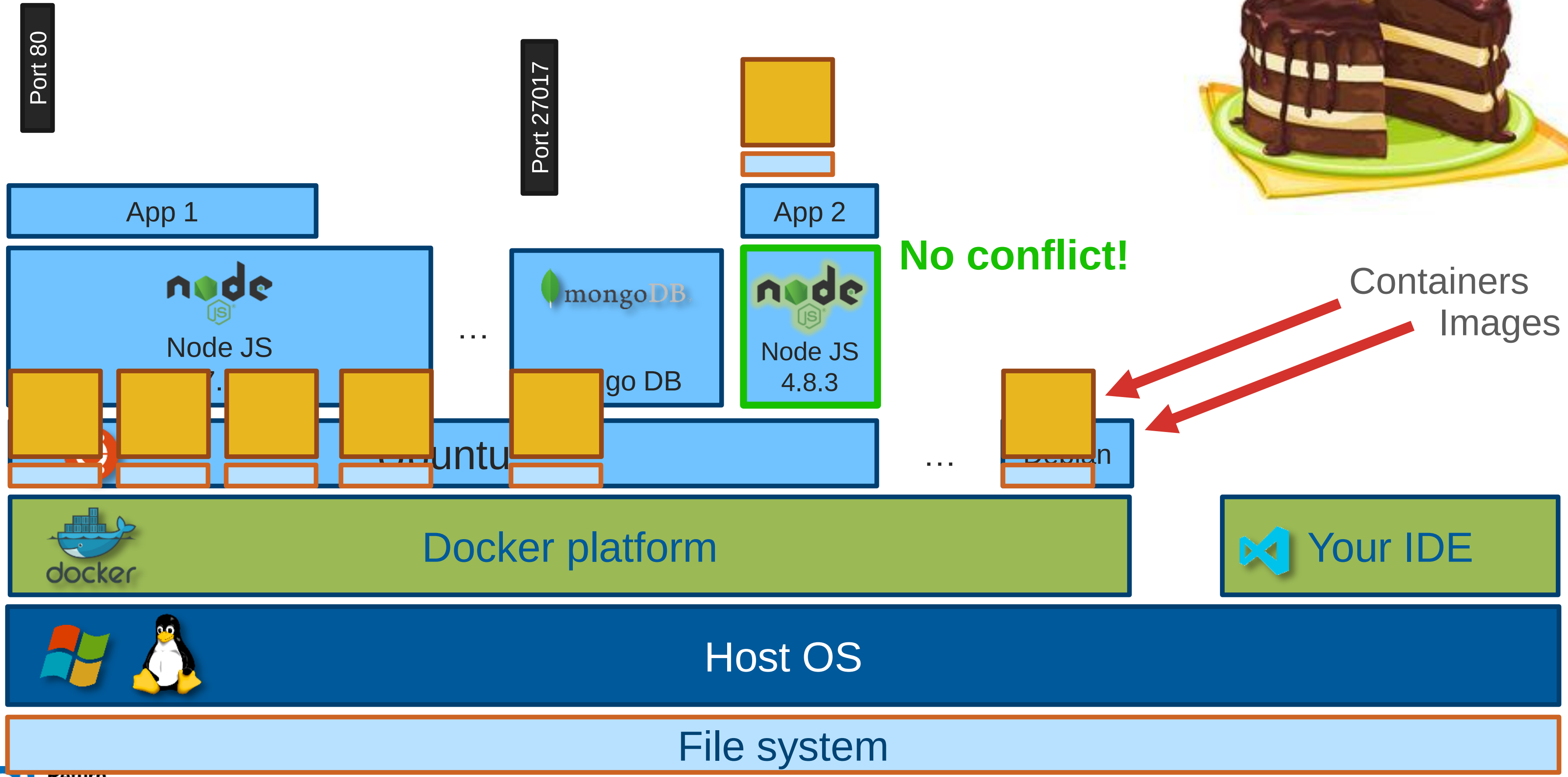
Bonus Goal:

4. Ready for **Cloud** and **Microservices**

Docker


Processes
Conflict!
Port 80
Port 27017
Executables
App 2
Node JS
4.8.3
App 1
Node JS
7.0
mongoDB
Mongo DB
Your IDE
Host OS
File system

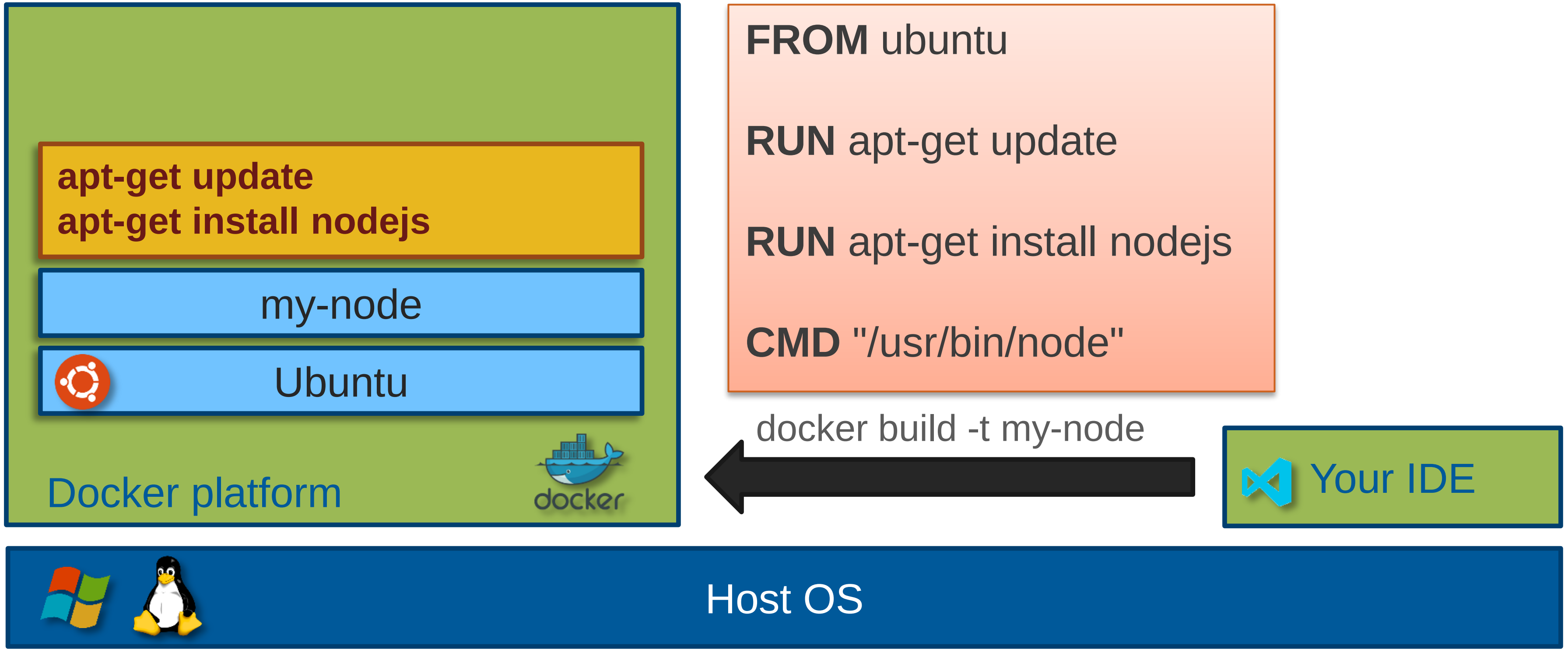
Docker


Port 80
Port 27017
App 1
App 2
No conflict!
Node JS
Node JS
4.8.3
mongoDB
go DB
…
untu
…
Containers
Images
docker
Docker platform
Your IDE
Host OS
File system

Docker

Building an image

Dockerfile



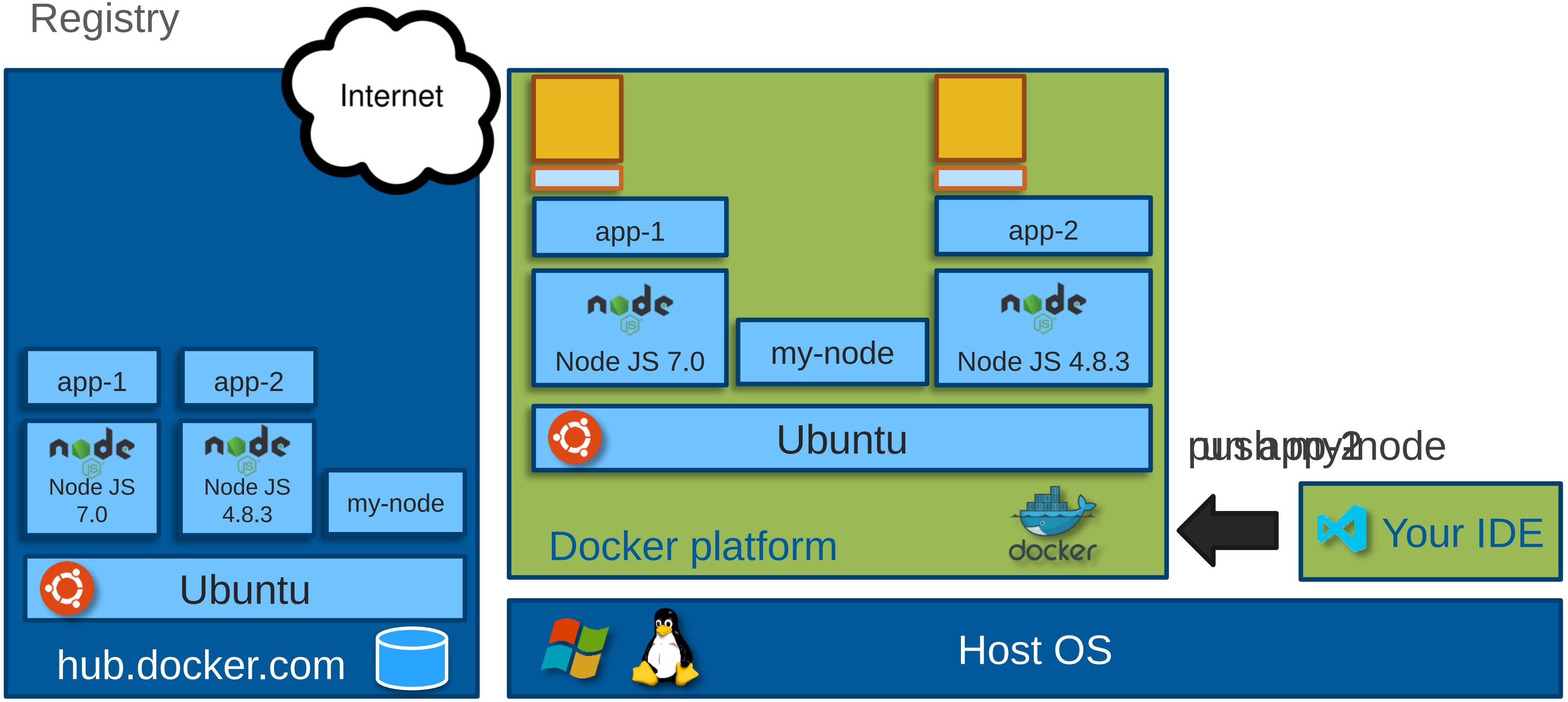
```
FROM ubuntu

RUN apt-get update

RUN apt-get install nodejs

CMD "/usr/bin/node"
```

```
docker build -t my-node
```

```
apt-get update
apt-get install nodejs
```

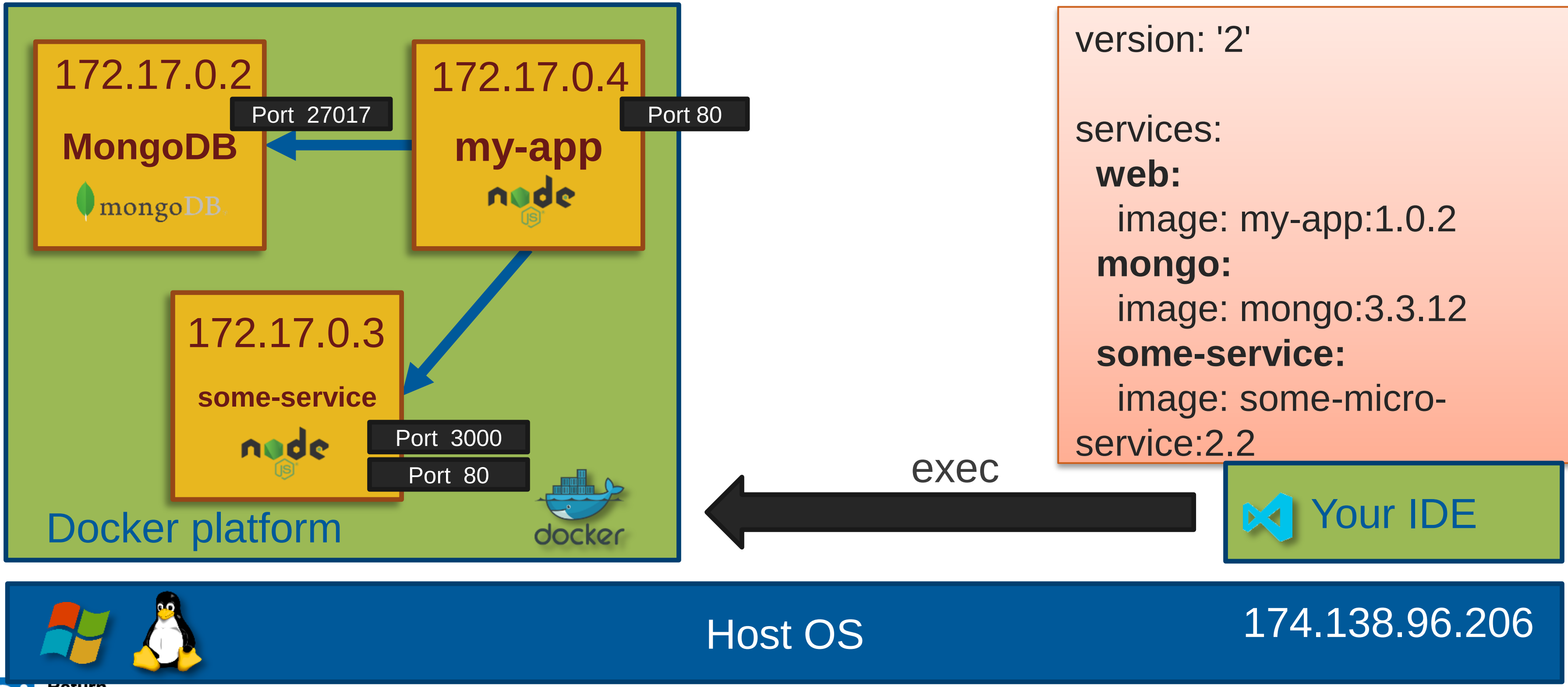
Docker

Registry



Docker

Multitier / Microservices



docker-compose.yml

```
version: '2'

services:
  web:
    image: my-app:1.0.2
  mongo:
    image: mongo:3.3.12
  some-service:
    image: some-micro-service:2.2
```

Return on Intelligence

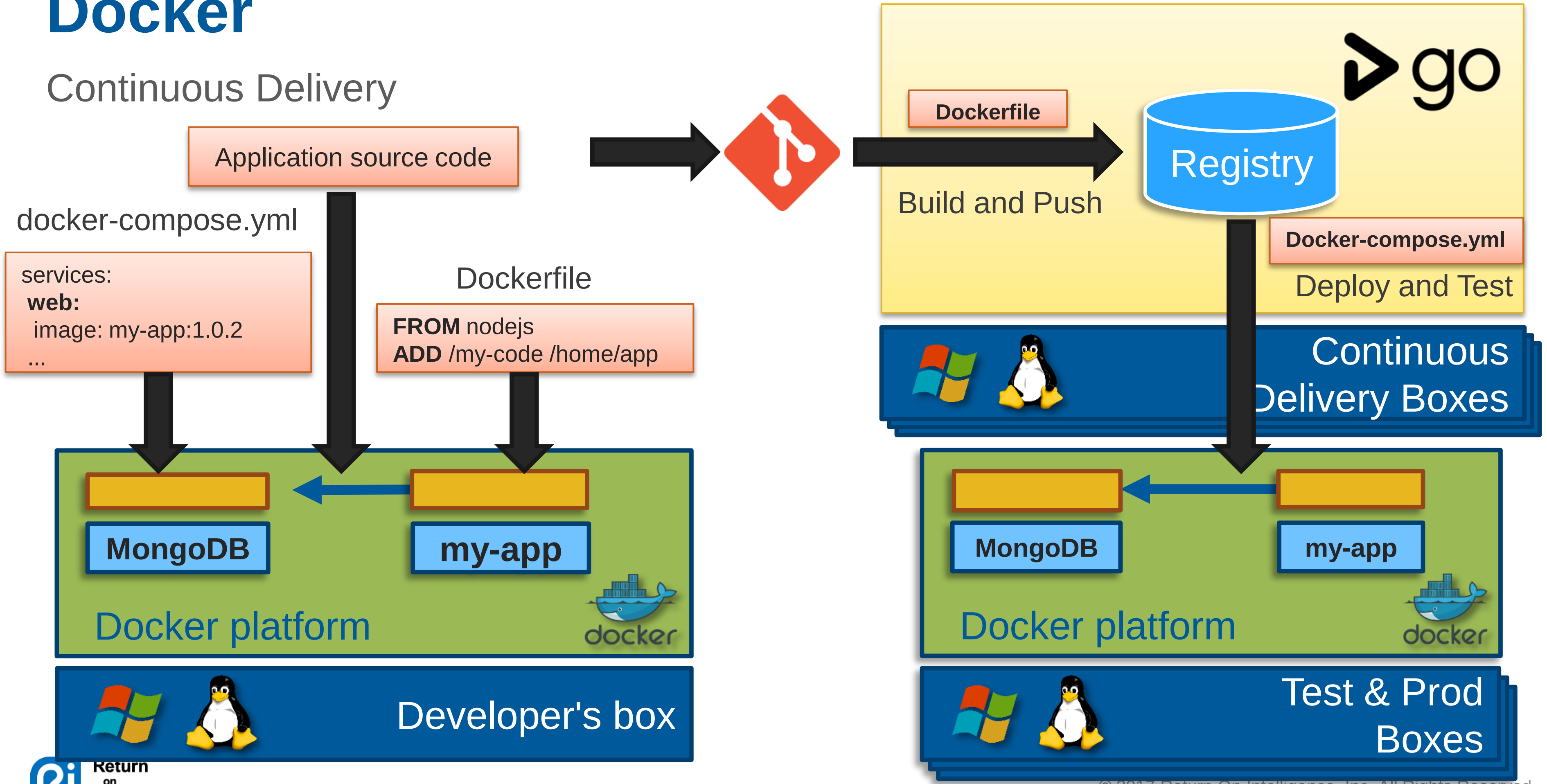
Docker

What solves & gives?

1. Environments identity → **Easy to provision prod like environments**
2. Dependencies next to code → **Simple deployment**
3. Efficient use of resources → **Fast deployment**

Docker

Continuous Delivery

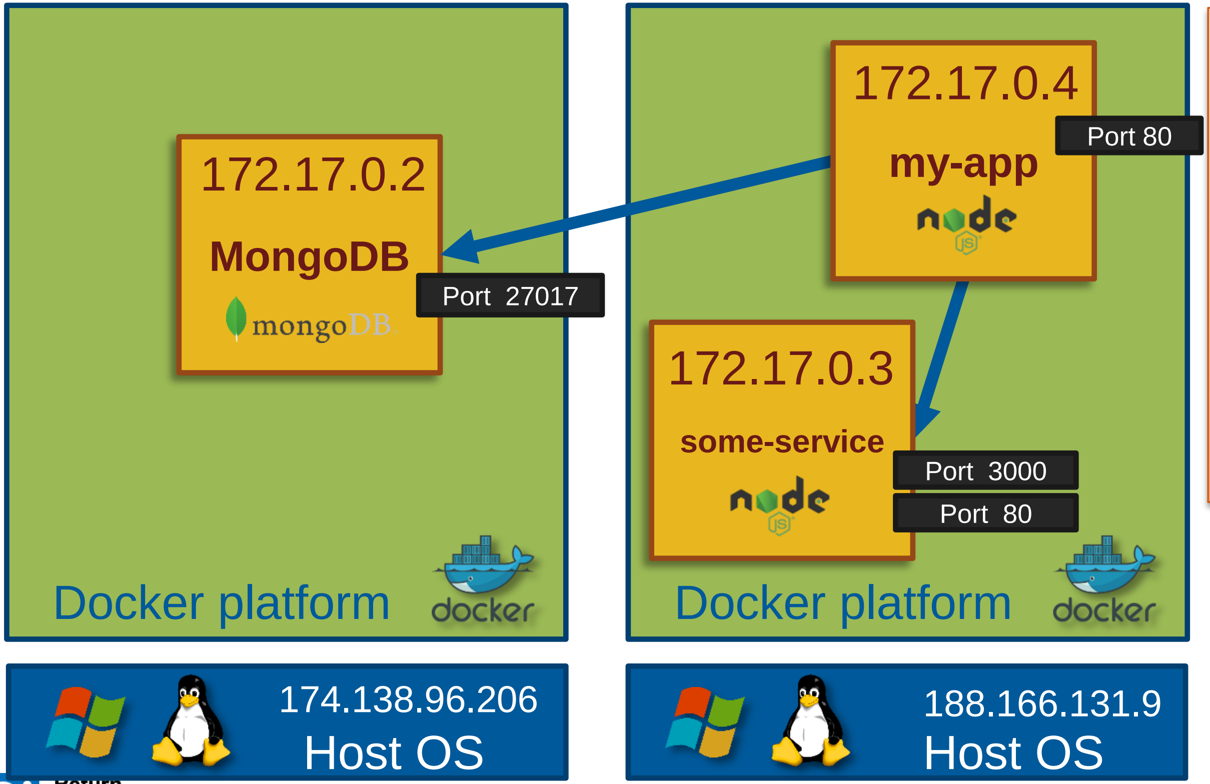


```
services:
 web:
  image: my-app:1.0.2
...
```

```
FROM nodejs
ADD /my-code /home/app
```

Rancher

Multihost?

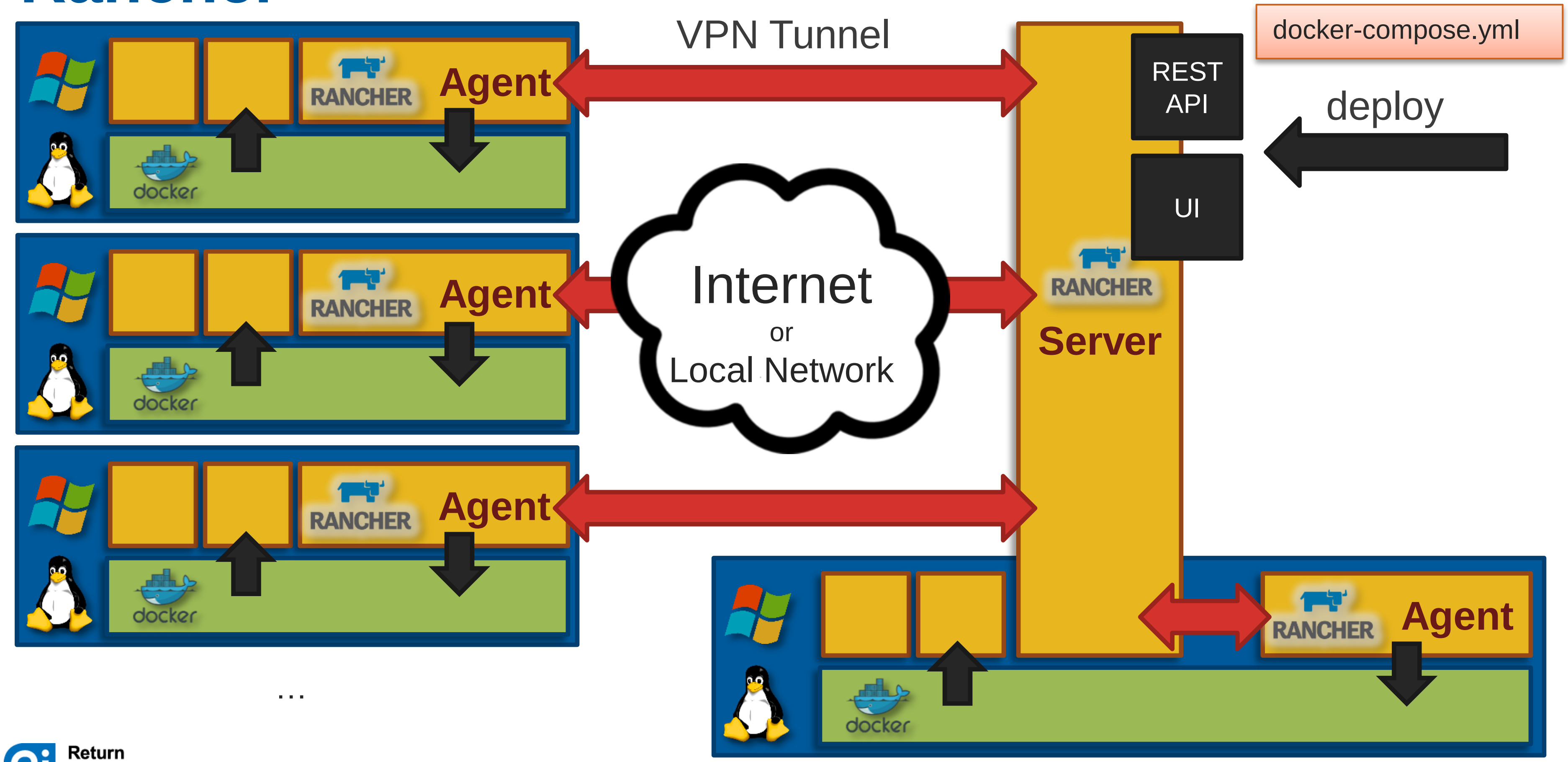


docker-compose.yml

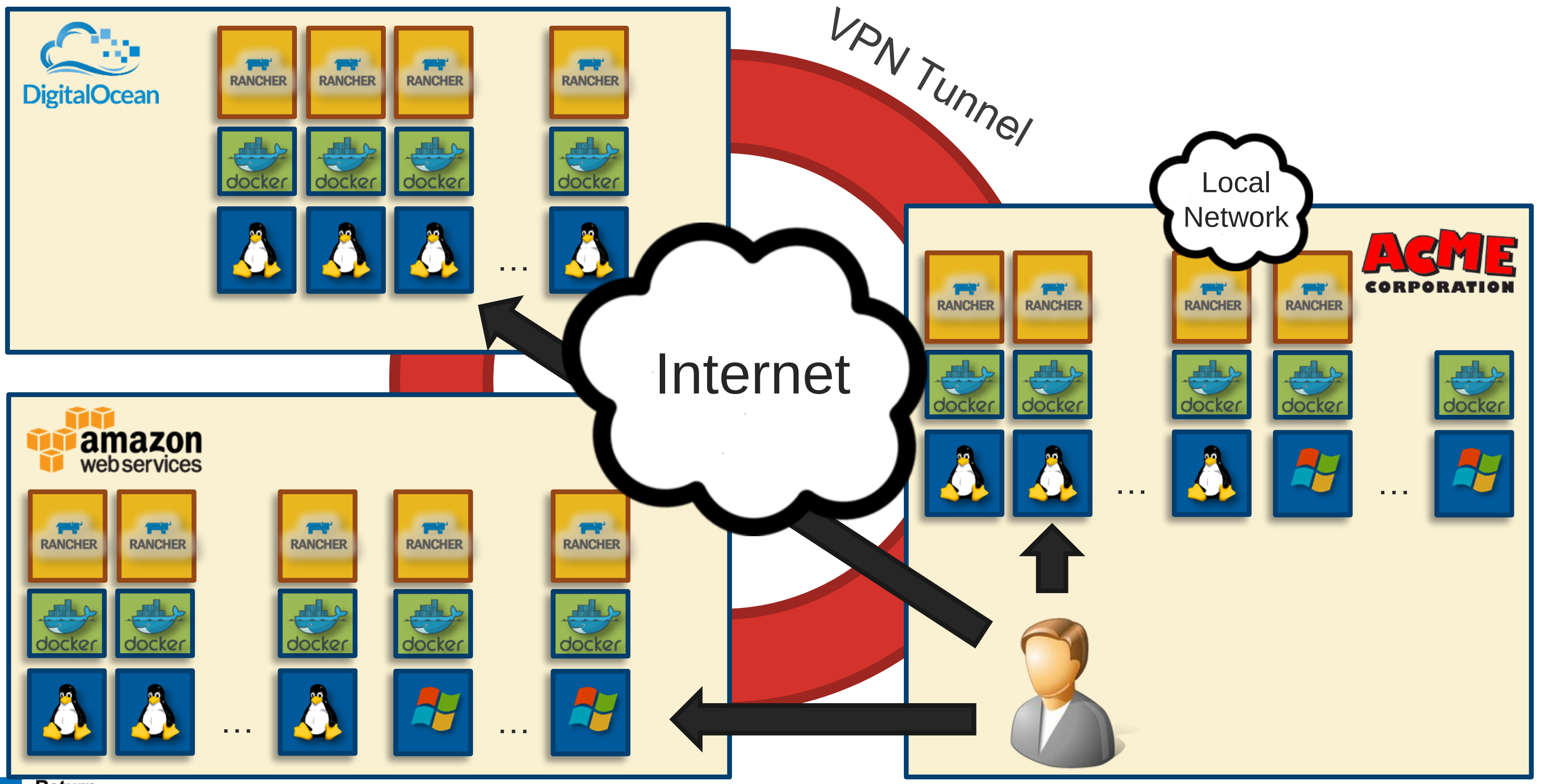
```
version: '2'
services:
  web:
    image: my-app:1.0.2
  mongo:
    image: mongo:3.3.12
  some-service:
    image: some-micro-service:2.2
```

deploy

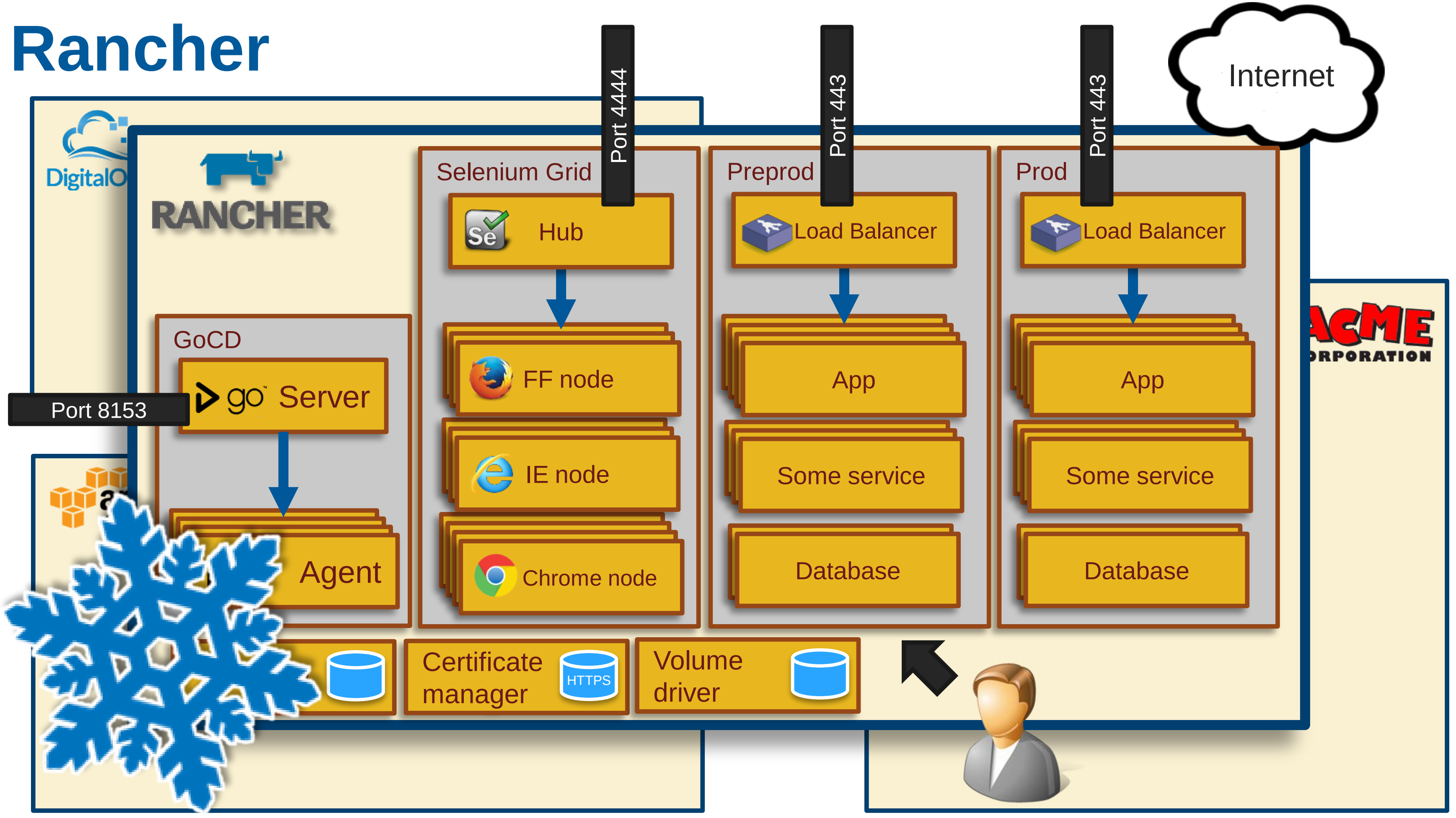
Rancher


VPN Tunnel
docker-compose.yml
REST API
deploy
UI
RANCHER
Agent
docker
Internet
or
Local Network
RANCHER
Server
RANCHER
Agent
docker
RANCHER
Agent
docker
RANCHER
Agent
docker
…

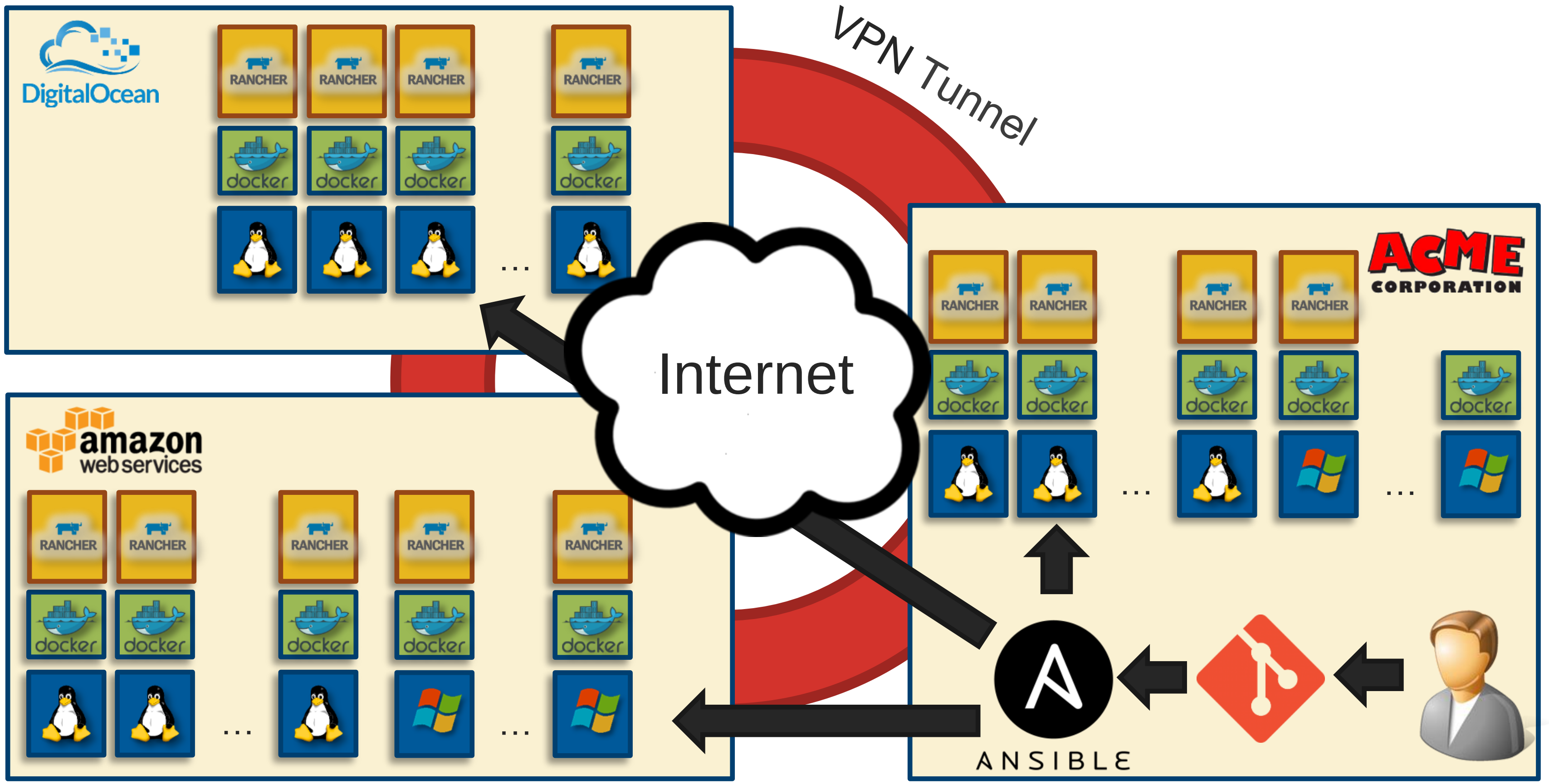
Rancher


DigitalOcean
RANCHER
docker
...
VPN Tunnel
Local Network
Internet
ACME CORPORATION
amazon web services

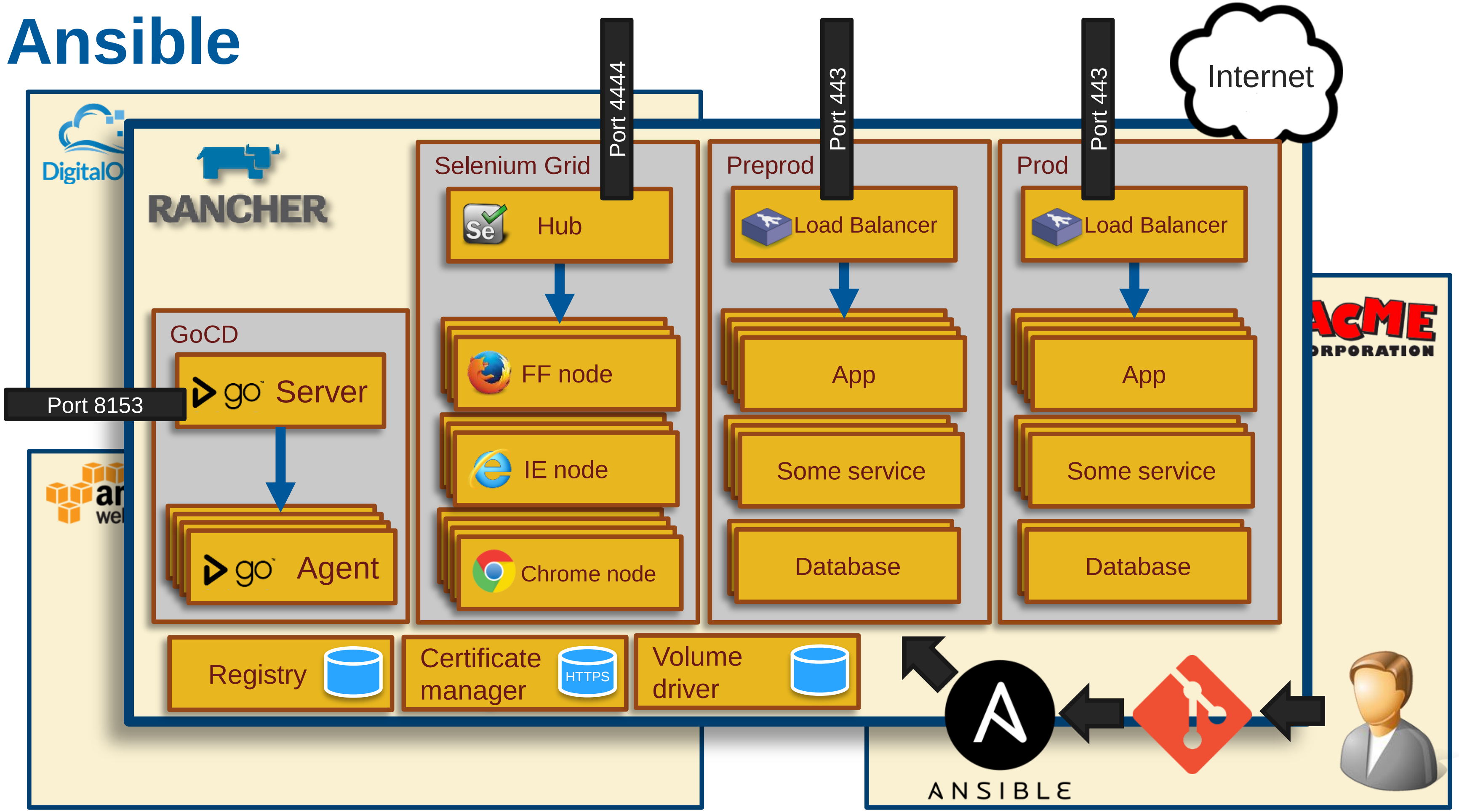
Rancher


Internet
Port 4444
Port 443
Port 443
DigitalO
RANCHER
Selenium Grid
Hub
FF node
IE node
Chrome node
Preprod
Load Balancer
App
Some service
Database
Prod
Load Balancer
App
Some service
Database
GoCD
go Server
Agent
Port 8153
Certificate manager
HTTPS
Volume driver
ACME
ORPORATION

Ansible


DigitalOcean
RANCHER
docker
...
amazon web services
VPN Tunnel
Internet
ACME CORPORATION
ANSIBLE

Ansible


Internet
Port 4444
Port 443
Port 443
DigitalO
RANCHER
Selenium Grid
Hub
FF node
IE node
Chrome node
Preprod
Load Balancer
App
Some service
Database
Prod
Load Balancer
App
Some service
Database
GoCD
go Server
go Agent
Port 8153
ar
wel
ACME
ORPORATION
Registry
Certificate manager
HTTPS
Volume driver
ANSIBLE

Demo

Return on Intelligence

Docker

What solves & gives?

1. Environments identity
2. Dependencies next to code
3. Efficient use of resources
4. Independent version upgrade/downgrade
5. No technology lockdown
6. Easy-to-make experiments
7. Easy-to-test
8. Feature-rich (volumes, networks, …)
9. Fast project member integration

Rancher

What solves & gives?

1. Mixed Cloud-Native Environment
2. Logical & physical scaling
3. Health-checks and recovery
4. Load balancer
5. Automatic DNS record management
6. Certificate management
7. Monitoring & Logging
8. Container management
9. Easy rollback
10. Blue-green deployments

Ansible

What solves & gives?

1. Reproducible infrastructure
2. Automatic provisioning
3. Version-controlled infrastructure
4. Audit

Thank You

We Appreciate Your Time

Questions?

Return on Intelligence

Contact Us

Sergey Gerasimov, Technical Leader

PR@returnonintelligence.com