Управление производственным процессом разработки программного обеспечения

Infrastructure as a Code (DevOps)

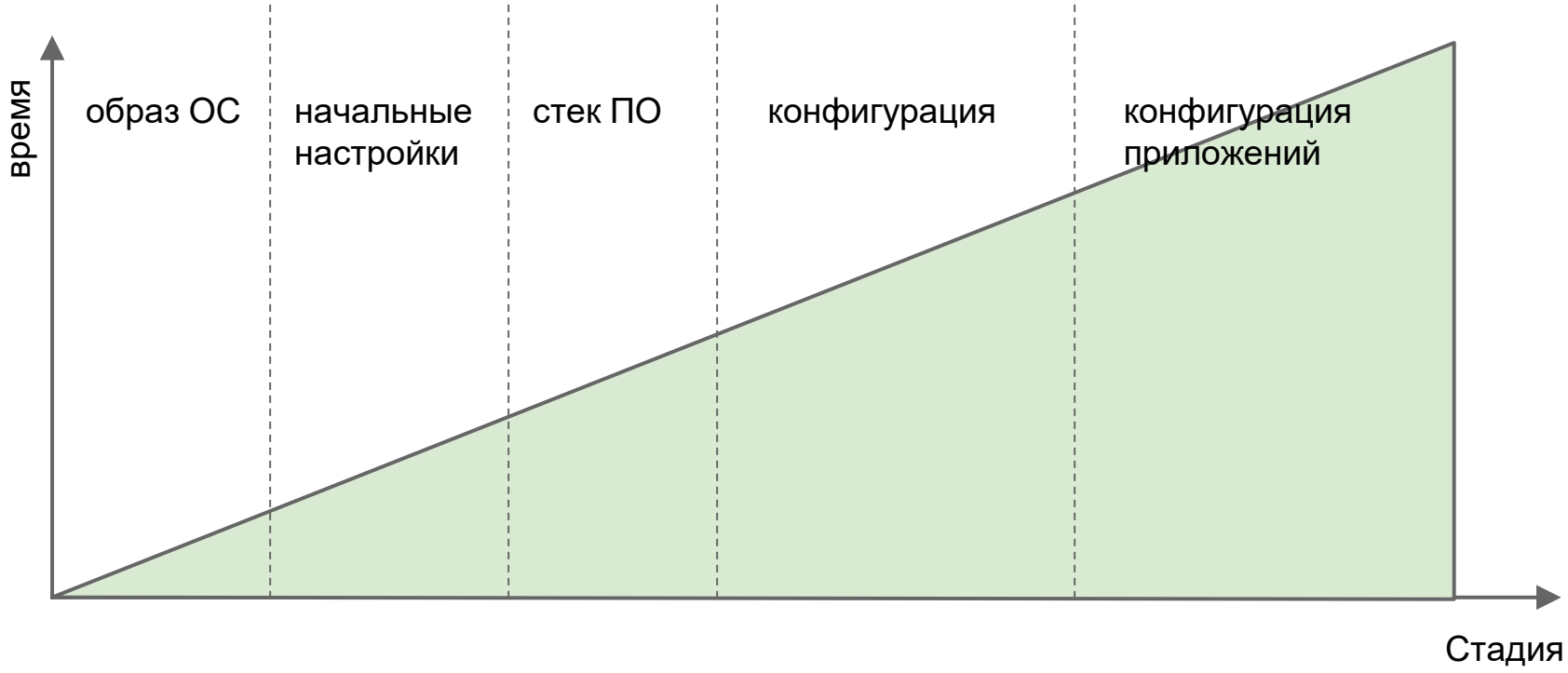
План

- Разработчики и администраторы
- История
- Архитектура
- Виртуализация

Configuration management

- Железо
- **Установленное ПО**
- **Настройки ПО**
- **Настройки сети**
- Мониторинг

Трудозатраты администратора



developers & operators

Dev/Ops

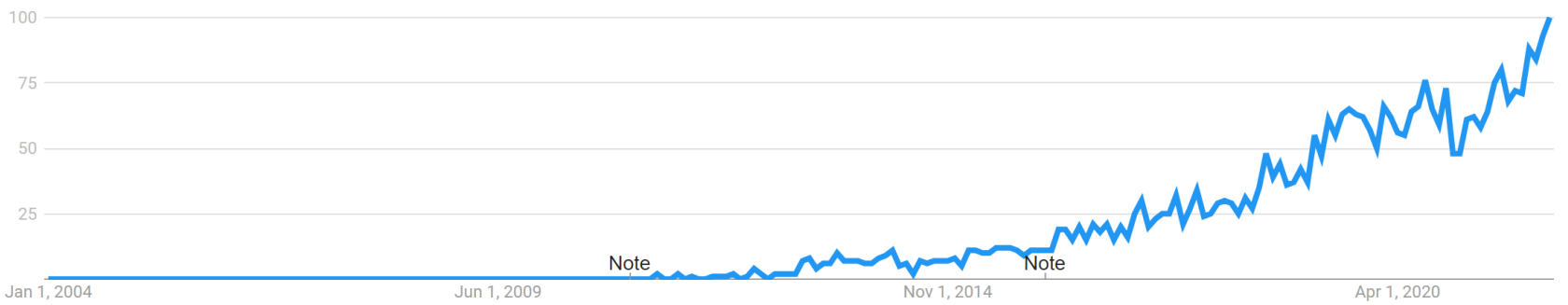
Google trend

devops
Search term

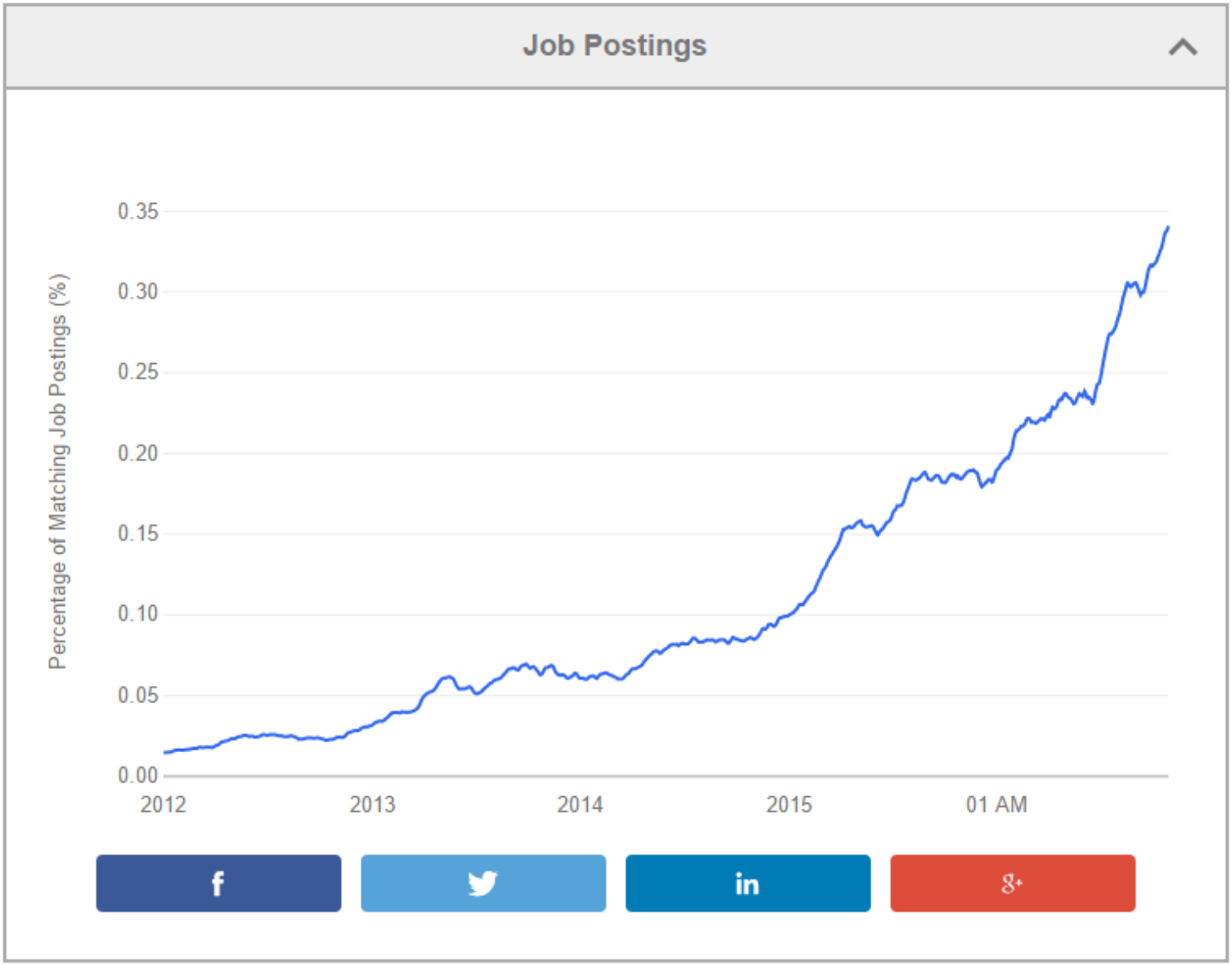
+ Compare

Russia | 2004 - present | All categories | Web Search

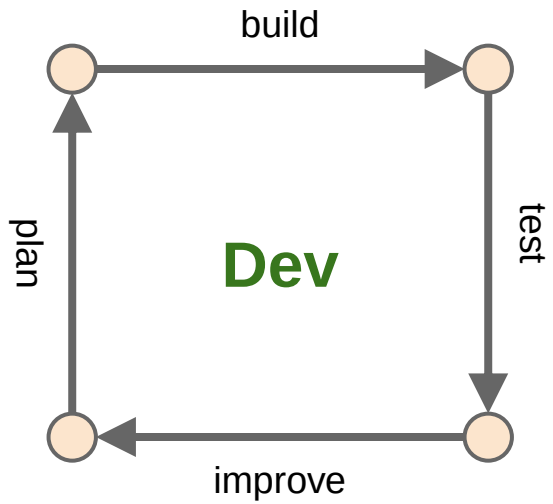
Interest over time



Indeed job trend by DevOps keyword



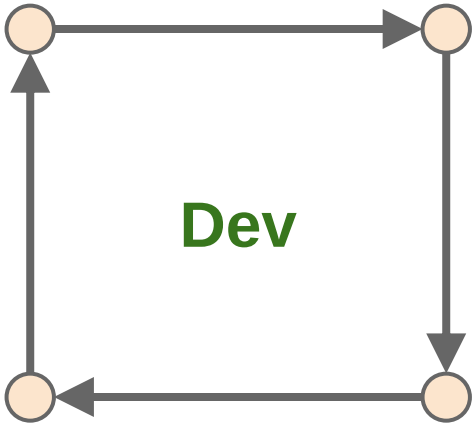
developers & operators

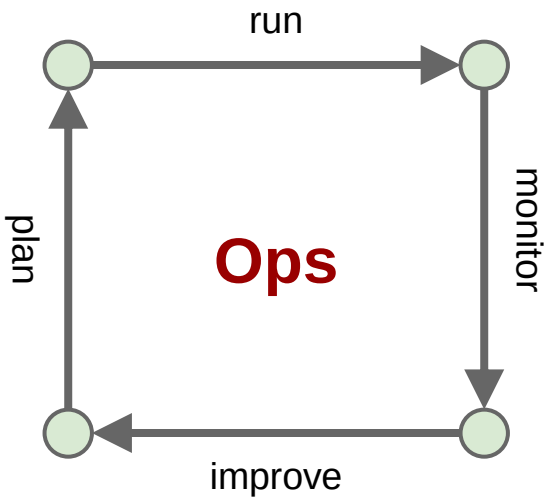


Ops

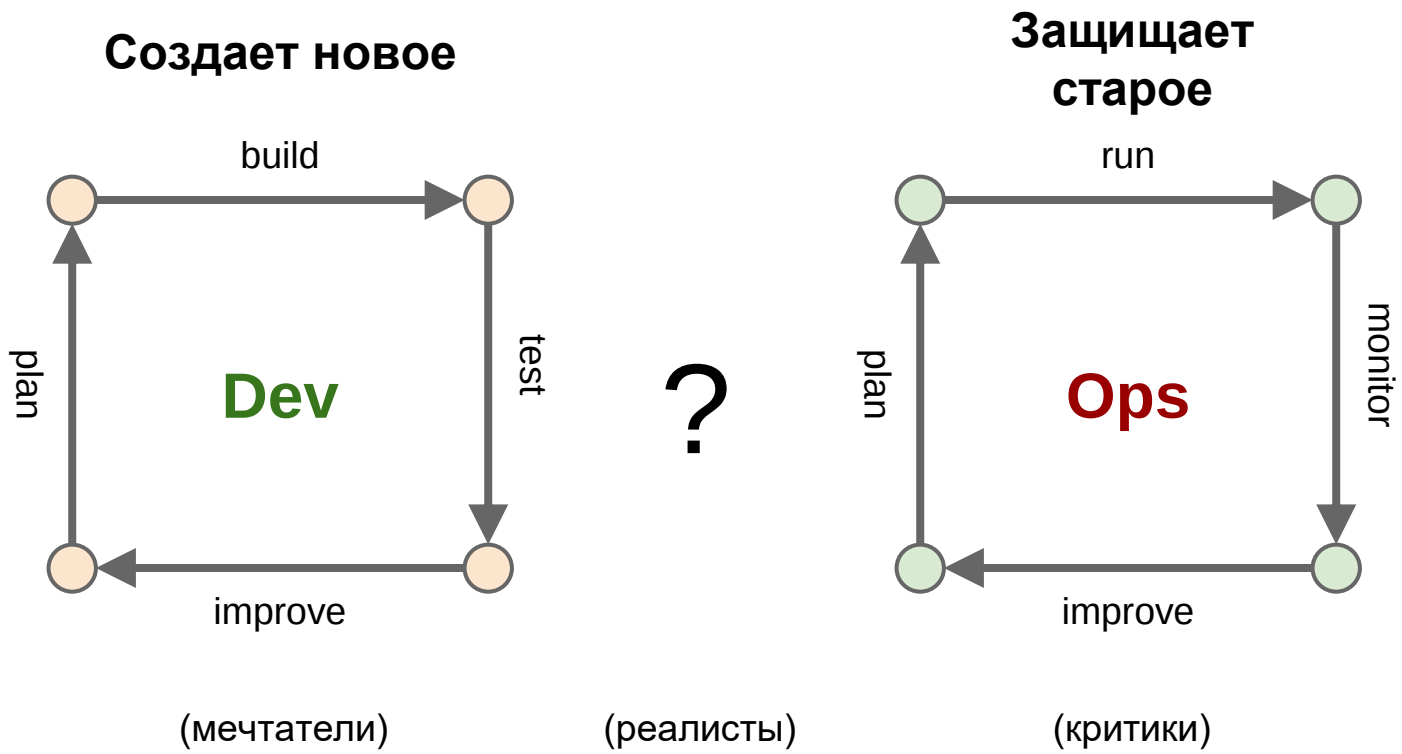
Люди: agile манифест
Процессы: XP, Scrum, Kanban
Инструменты: юнит тесты, непрерывная интеграция

developers & operators

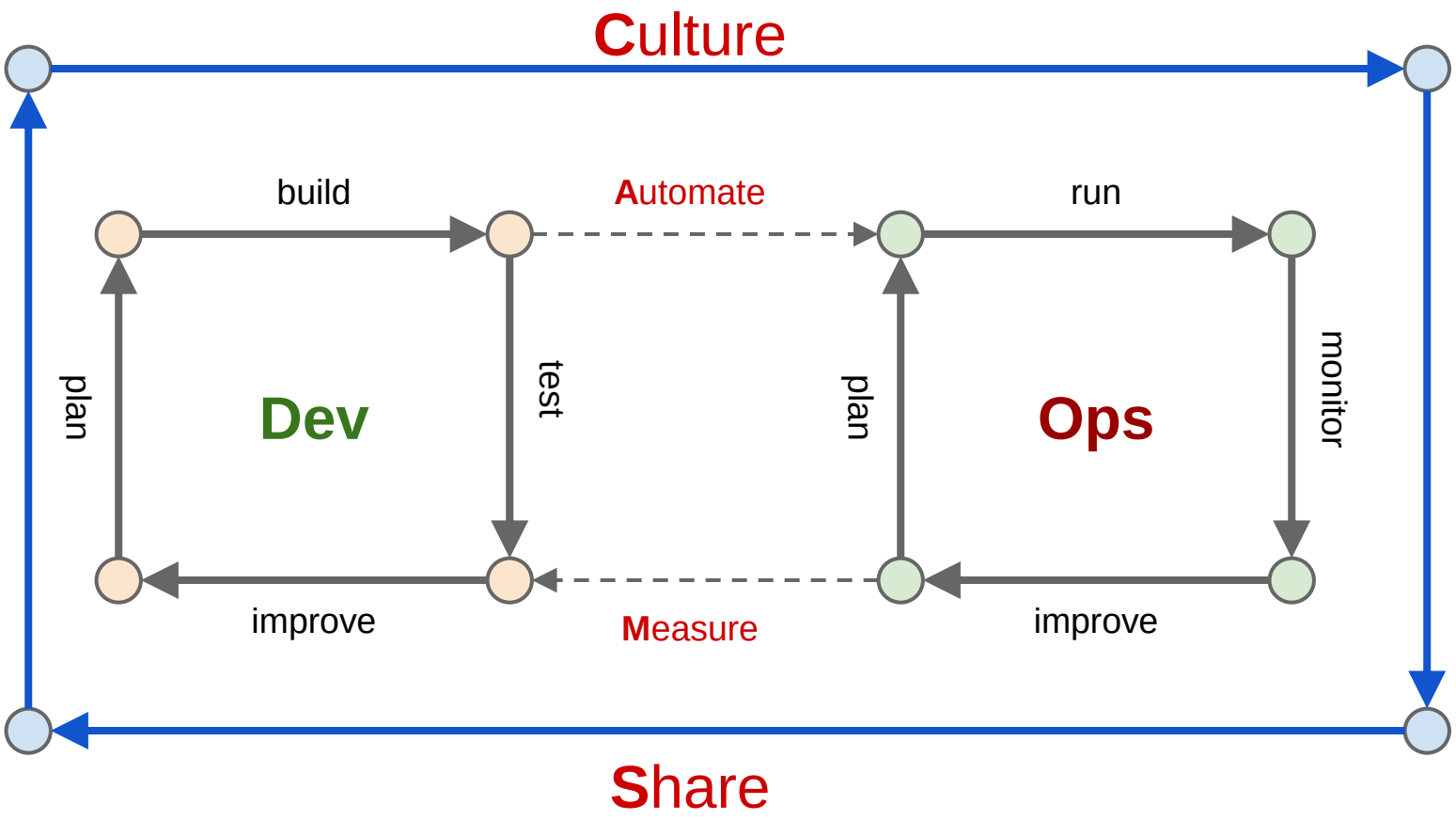




Обратная связь



CAMS


Culture
build
Automate
run
plan
Dev
test
plan
Ops
monitor
improve
Measure
improve
Share

Culture

Люди и процессы на первом месте. Без культуры попытки все автоматизации провалятся.

Automation

- Инфраструктура как код
- Версионирование
- Переиспользование конфигураций (dev, test, prod)
- Визуализация (с точки зрения управления)

Measure

- Доступ к логам и метрикам
- Расследование инцидентов совместно с разработчиками
- Определение повторяющихся проблем

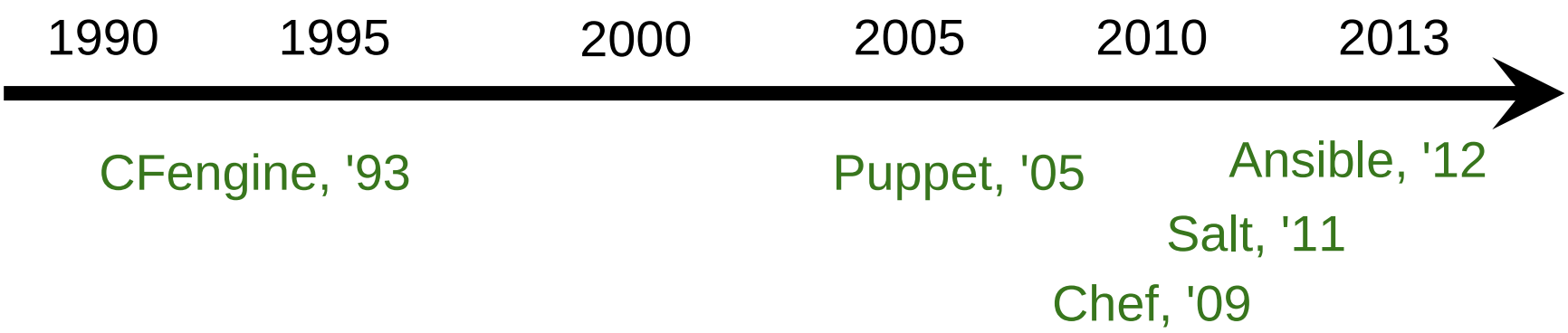
Share

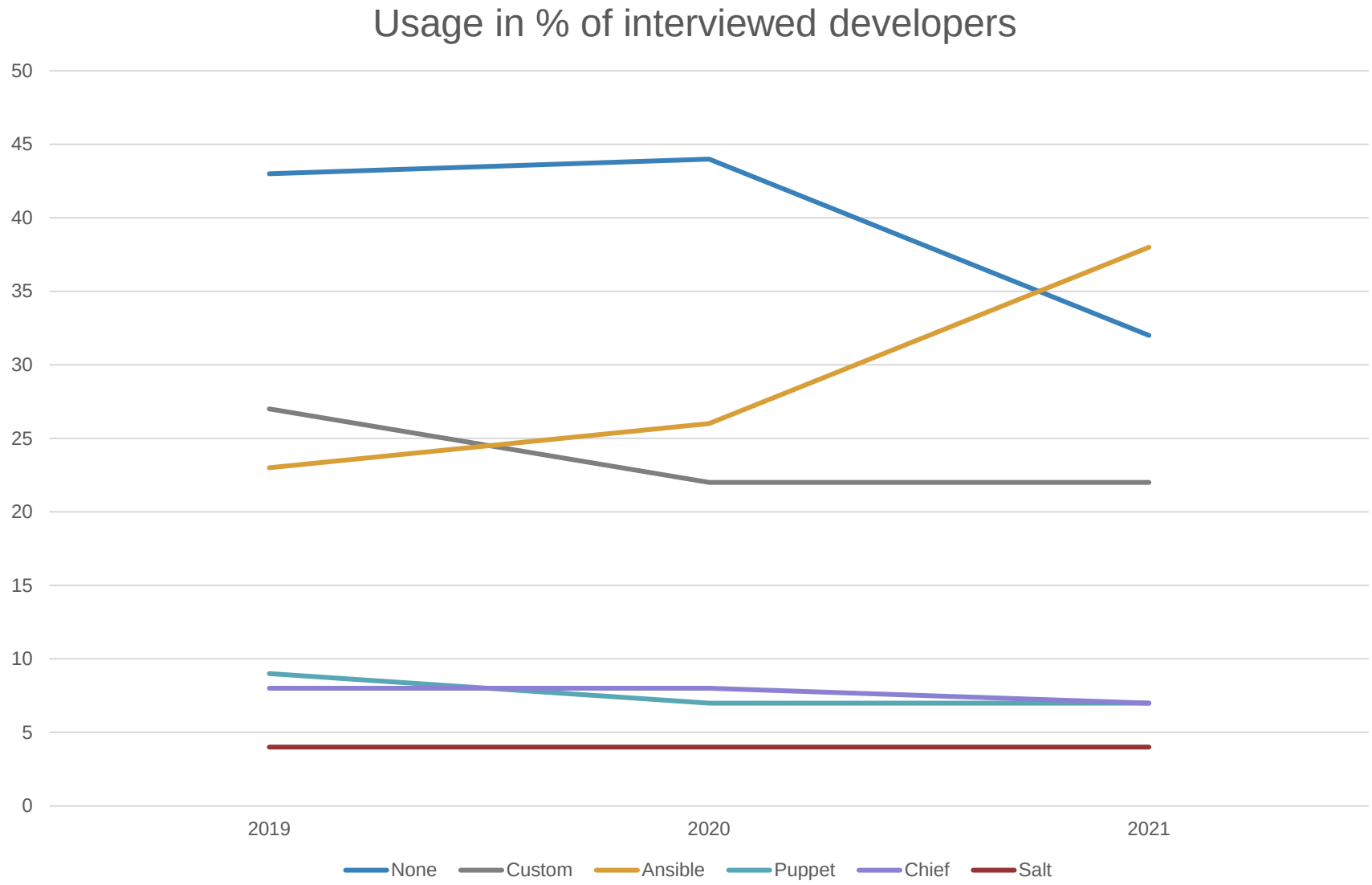
- От разработчиков к операторам (разработчики вовлечены в поддержку рабочего окружения)
- От операторов к разработчикам (обратная связь об ошибках)

Continuous Integration Maturity Model

Initial	Процессы непредсказуемые, слабо контроллируемые, реактивные	VCS, ночные сборки, багтрекер
Managed	Процессы на уровне проектов, реактивные	Build on commit, aut. deploy to dev, unit tests, static code analysis, documentation generation
Defined	Процессы на уровне организации, проактивные	aut. deploy on test, on demand deploy to prod, project metrics
Quantitatively Managed	Процессы измеряются и контроллируются	aut. func. tests, aut. deploy to prod, environment metrics, alerts
Optimizing	Фокус на совершенствовании процессов	cont. deploy to prod.

Configuration Management



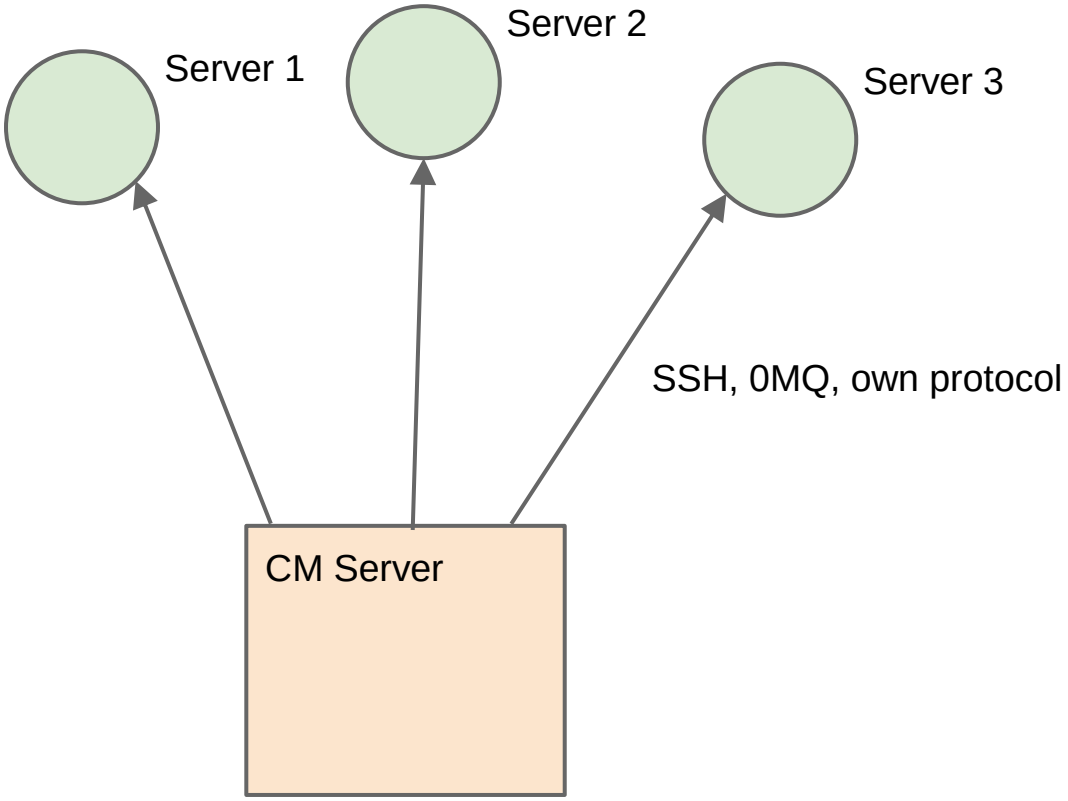


Источник: https://www.jetbrains.com/lp/devecosystem-2021/devops/

Общие черты

- push/pull модель
- DSL, часто на ruby
- Переиспользуемые рецепты (recipe, playbooks)
- Декларативный стиль (мы описываем что хотим получить в итоге, а как этого достичь решает инструмент)

push



Chef-solo, Ansible

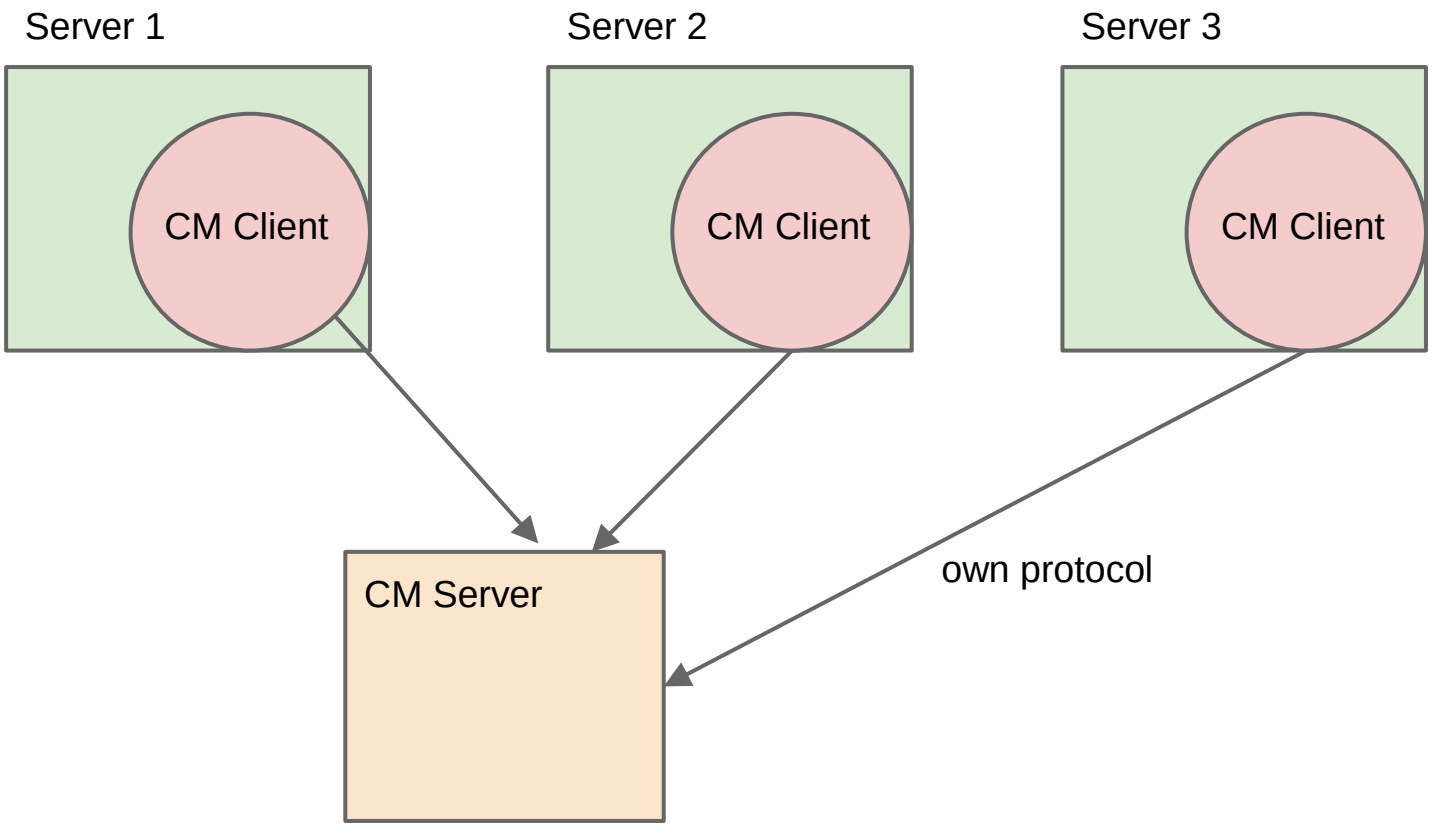
Достоинства Push модели

- Не требует установки дополнительного ПО на серверах
- Более привычно для пользователей Linux

Недостатки Push модели

- Зависит от качества соединения
- Требует sudo
- Требует доступа на все управляемые сервера

pull



Chef, Puppet, Salt, Ansible, CF engine

Достоинства Pull модели

- Централизованное управление
- Горизонтальное масштабирование
- Не требует дополнительных пользователей на сервере

Недостатки Pull модели

- Требует первоначальной установки клиента
- Централизованное управление
- Непрозрачно для пользователей

Ansible facts

- Новый (первый релиз 2012)
- написан на Python
- Простой (годится даже для разработчиков)
- Push based
- Работает через SSH
- Использует YAML в качестве языка рецептов
- one-man software

Ansible example

setup-user-env.yml

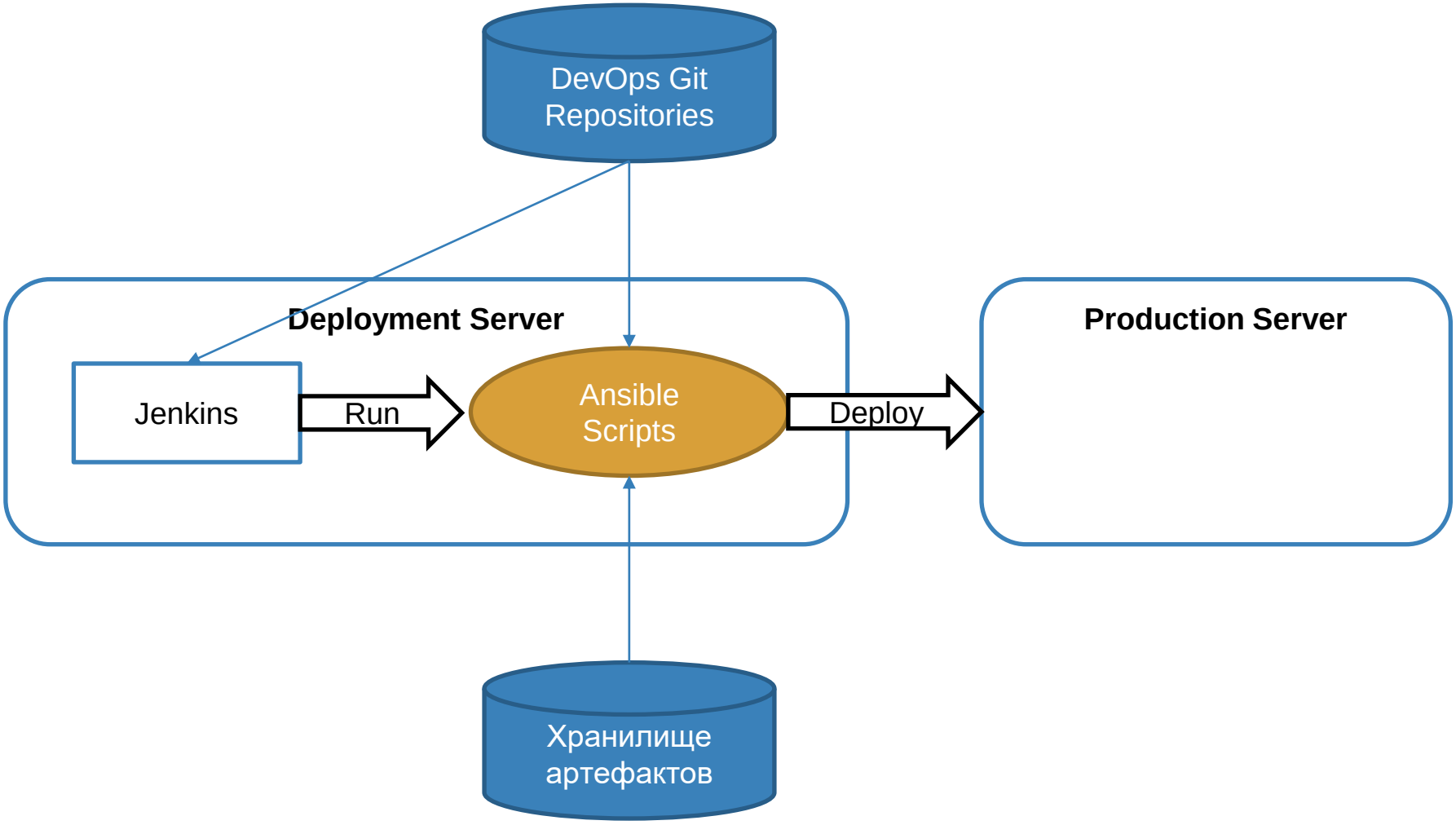
```
---
- hosts: ubuntu
  sudo: yes
  tasks:
    - name: Upload .bashrc file with colorful prompt
      action: copy src=files/bash/.bashrc dest=~/.bashrc
    - apt: pkg=vim
    - apt: pkg=iotop
    - apt: pkg=htop
    - apt: pkg=curl
```

hosts.list

```
[ubuntu]
dev001.example.com
dev002.example.com

#ansible-playbook --inventory-file=hosts.list setup-user-env.yml
```

Ansible example

```
- hosts: '{{ hosts }}'
  user: ansible
  become: true
  max_fail_percentage: 10

  vars:
    frontendOnly: false
    serialNumber: "{{ serialNumberSet | default(2) }}"
    javaHome: "/usr/java/jdk-15.0.2"

  serial: '{{ serialNumber }}'

  roles:
    - { role: node-exporter }
    - { role: aerospike-bidder }

  tasks:
    - name: Prepare directory /usr/java
      file:
        path: '/usr/java'
        state: directory
        owner: root
        group: root
        mode: 0755

    - name: Get OpenJDK
      get_url:
        url: 'https://personartrb-files.s3.amazonaws.com/openjdk-{{ java_version | default("15.0.2")
}}_linux-{{ java_arch | default("x64") }}_bin.tar.gz'
        dest: '/tmp/java.tar.gz'
        mode: 0444
        force: yes
```


DevOps Git
Repositories
Deployment Server
Jenkins
Run
Ansible
Scripts
Deploy
Production Server
Хранилище
артефактов

Salt facts

- Сравнительно новый (2011)
- Написан на Python
- Push based
- Для рецептов используется YAML
- Простой

Chef facts

- Зрелый (2009)
- написан на Ruby + Erlang
- Для рецептов используется DSL на Ruby
- pull/push based
- Hosted chef (платный)
- Отличная библиотека рецептов
- Огромное сообщество
- Сложный

Puppet facts

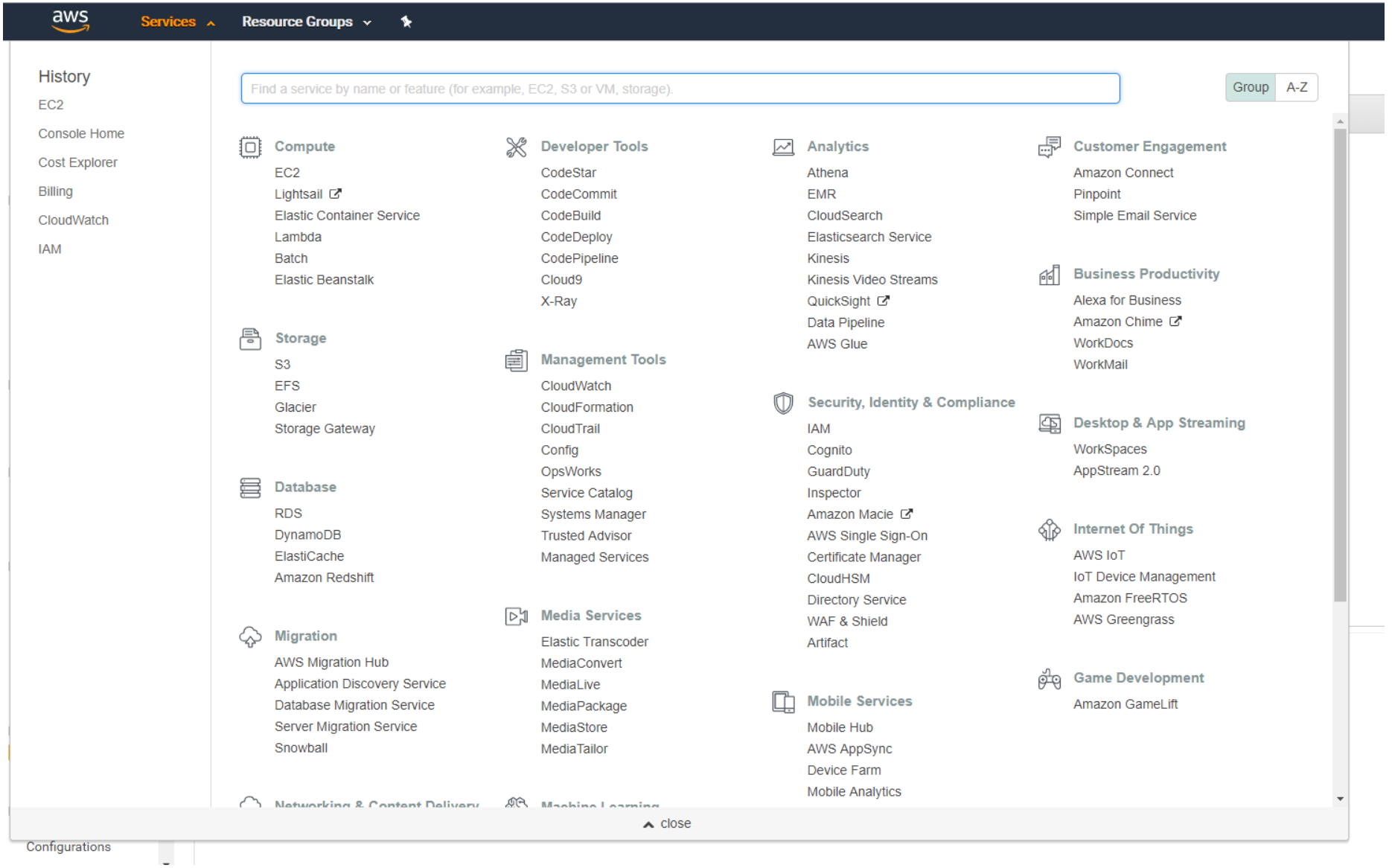
- Старый и зрелый (2005)
- Написан на Ruby
- DSL на Ruby для рецептов
- Огромная библиотека модулей
- Push based
- Есть коммерческая поддержка
- Сложный

Amazon Web Services (AWS)

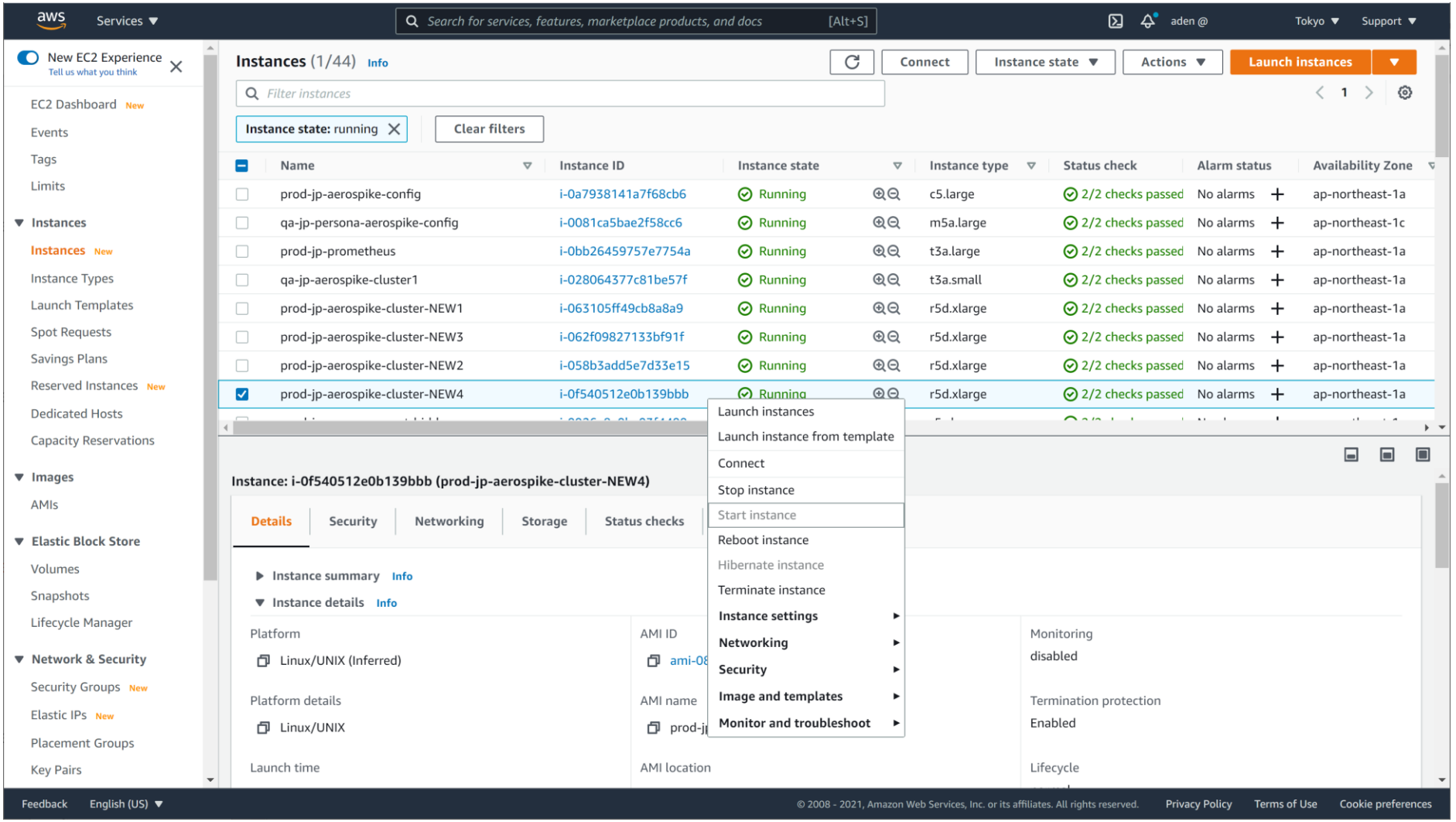
Проблема:
Большие проекты требуют сложного окружения
Решение:
Виртуальные машины в облаке
Дополнительно:

- Несколько дата-центров
- *RDS – Relational Database Service*
- *S3* – Simple Storage Service
- *Redshift* - data warehouse service
- *Batch* – пакетная обработка данных
- и многое другое …

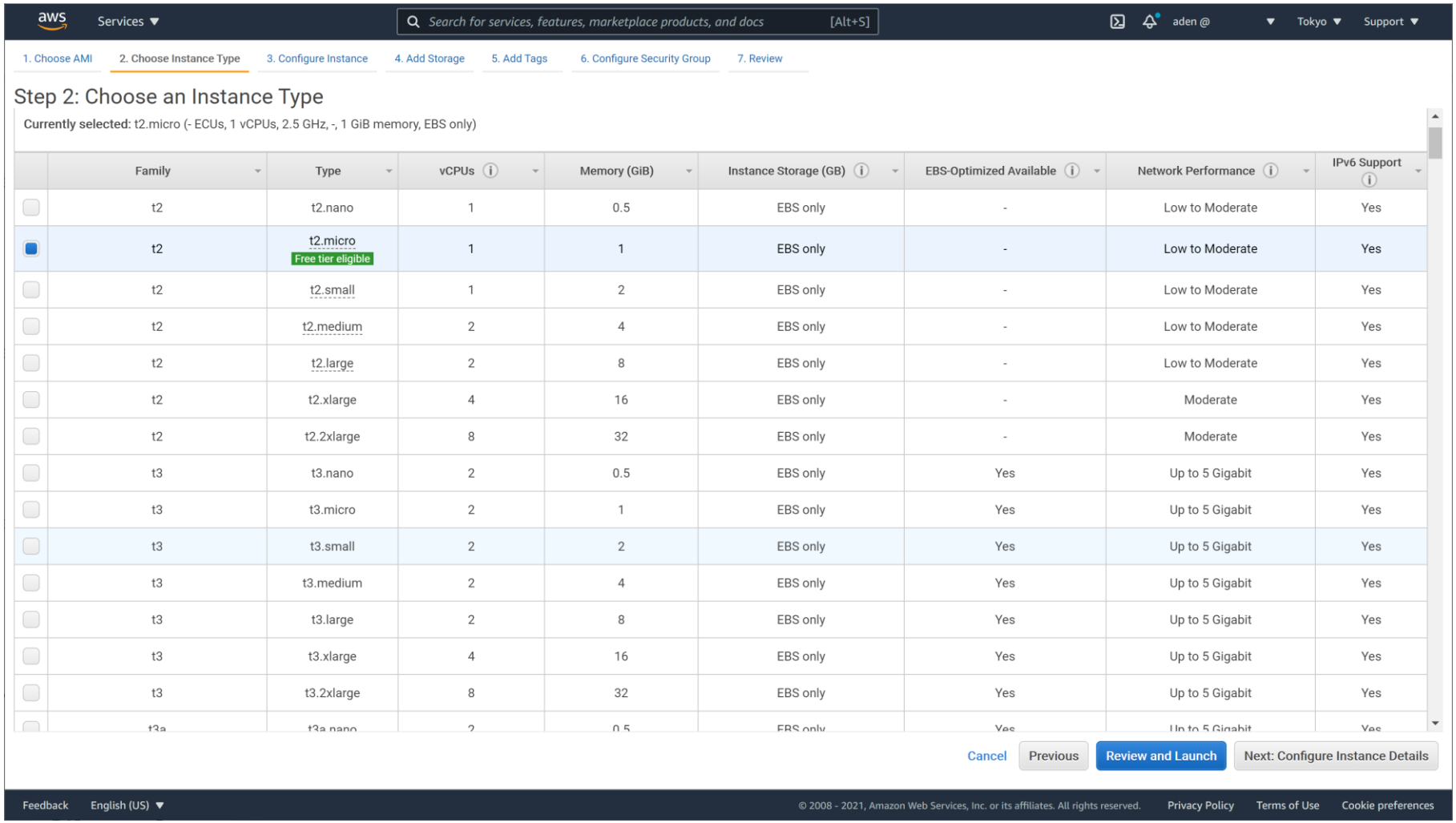
Amazon Web Services (AWS)



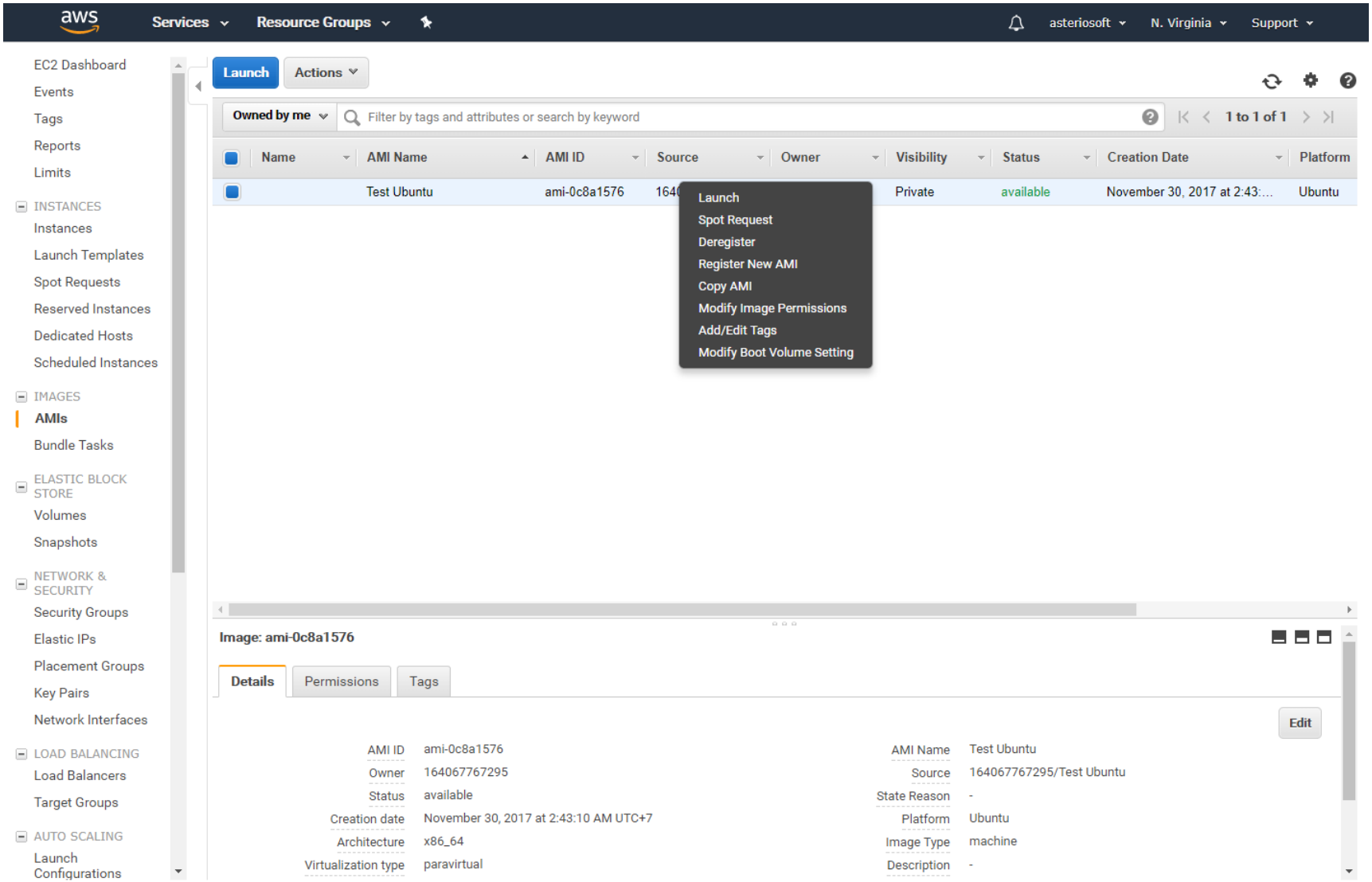
AWS Management Console



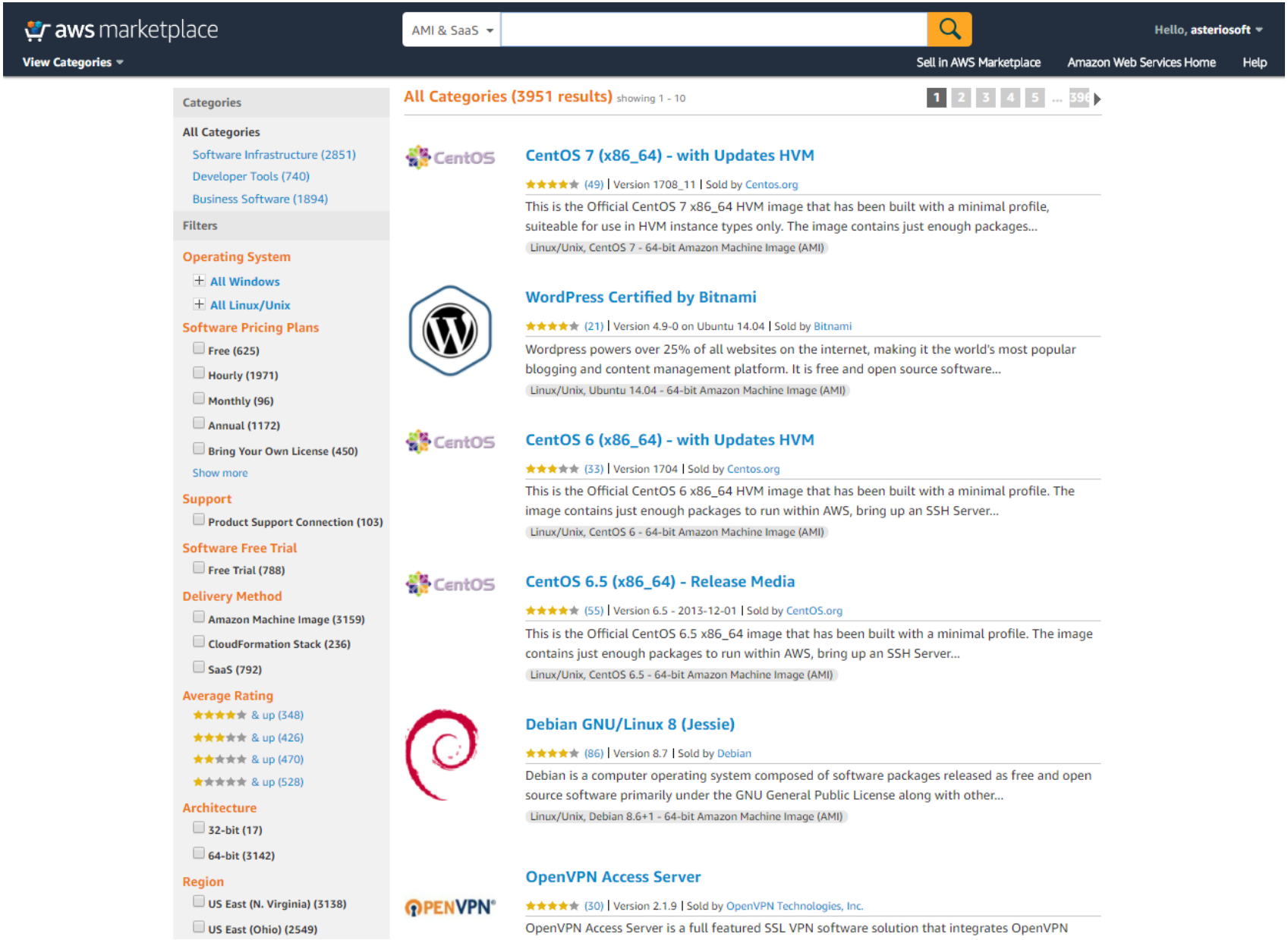
AWS New Instance Launch



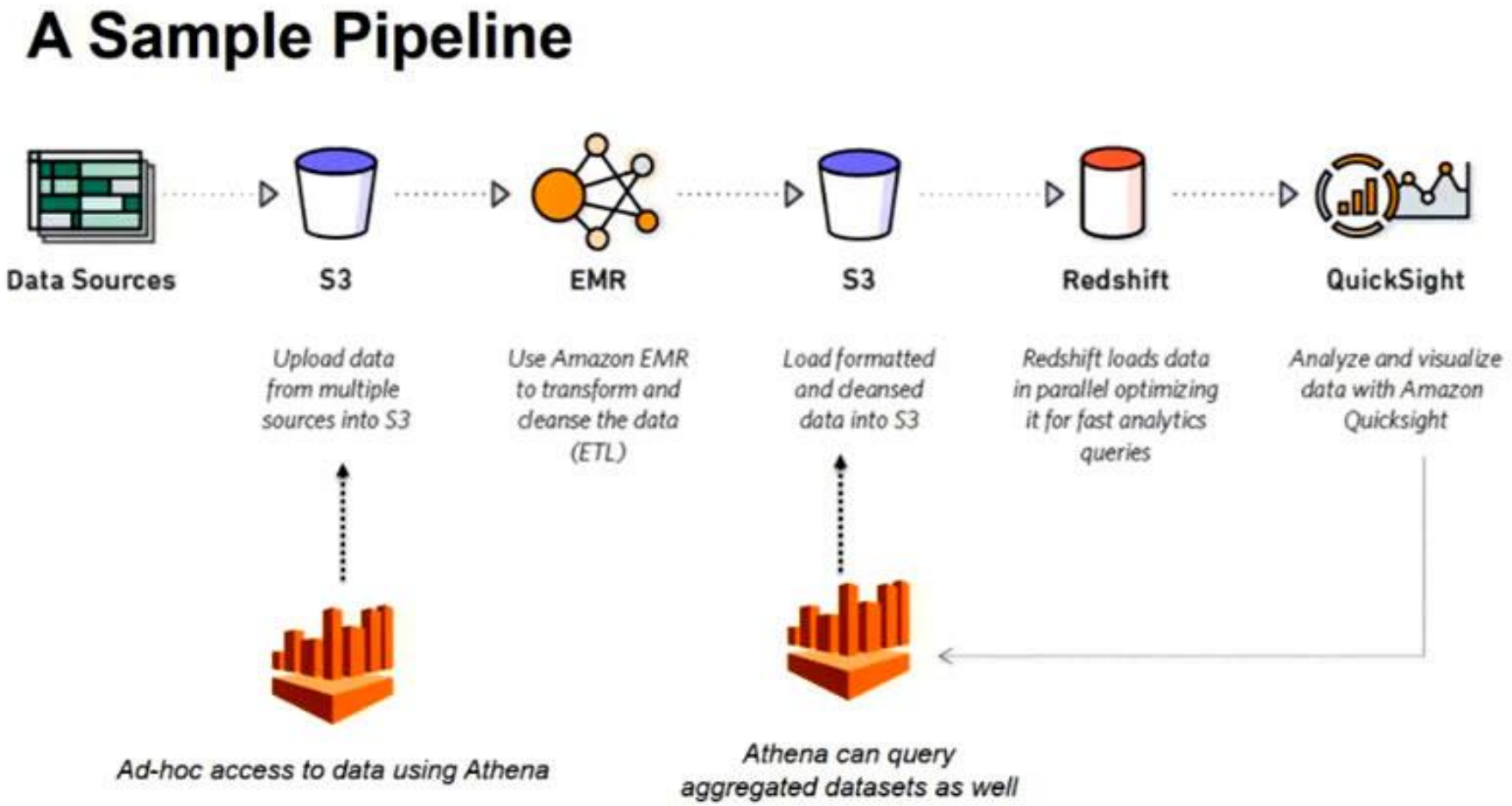
AMI – Amazon Machine Images



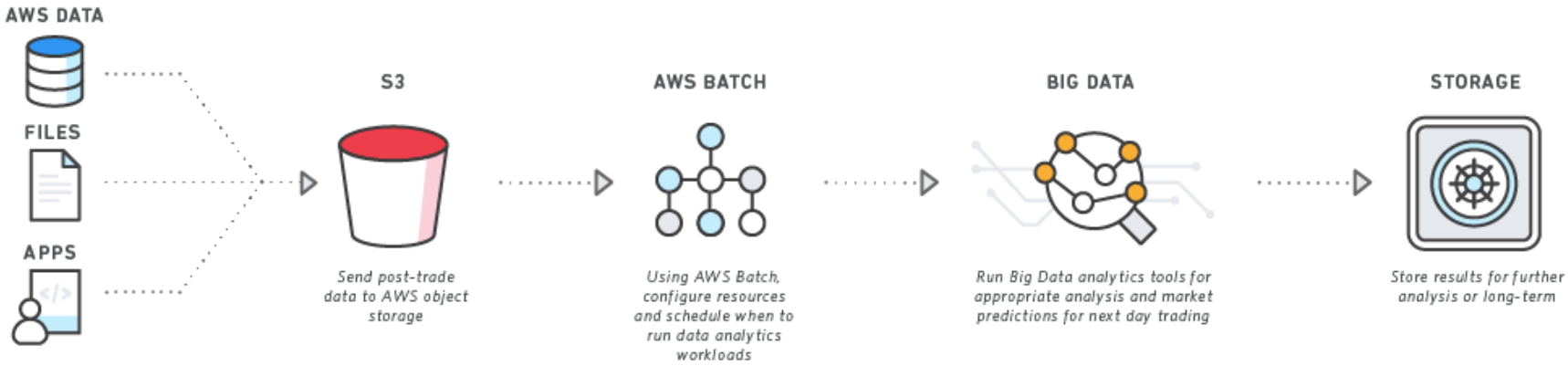
AWS Marketplace



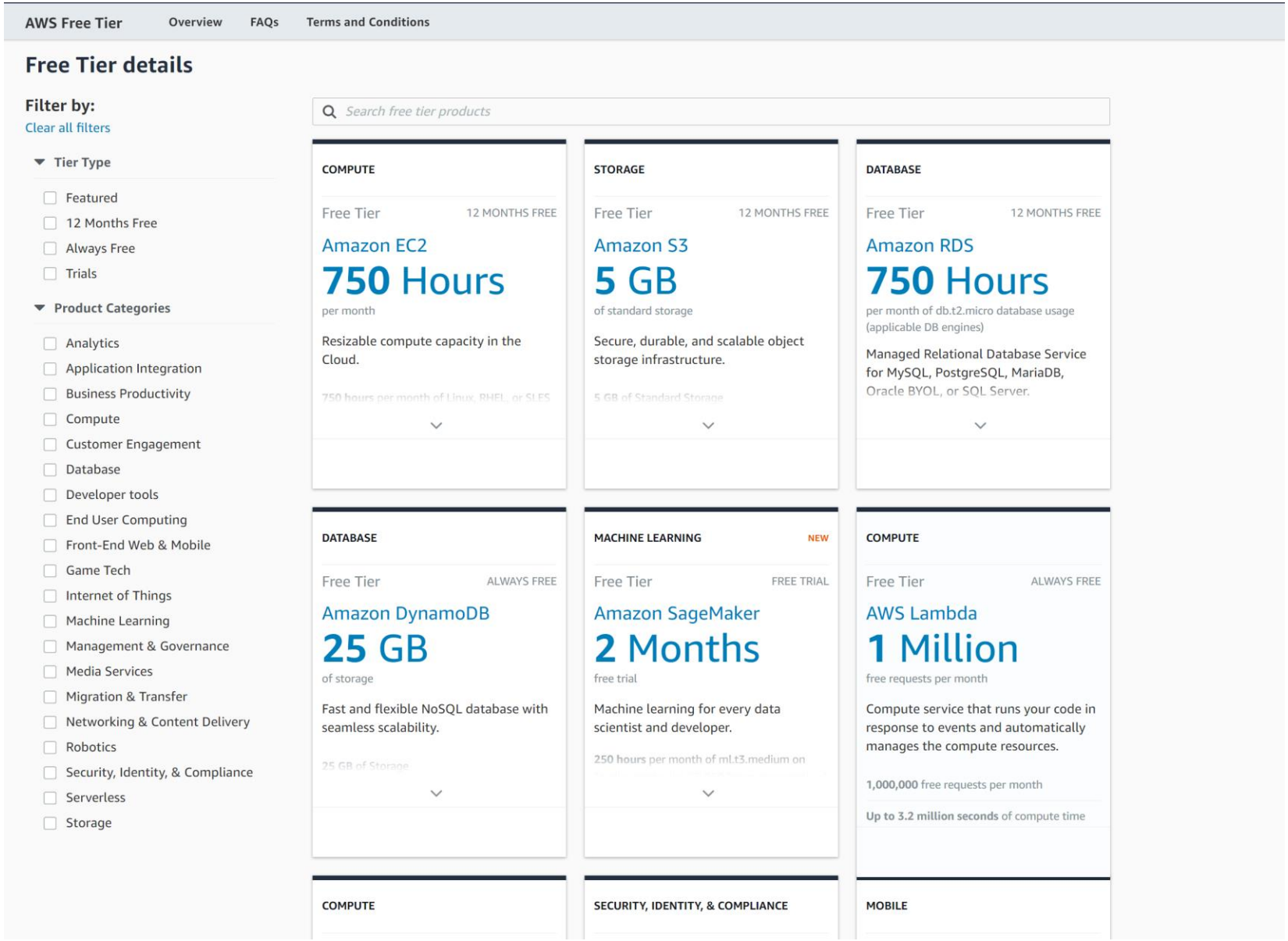
AWS Athena & AWS Redshift



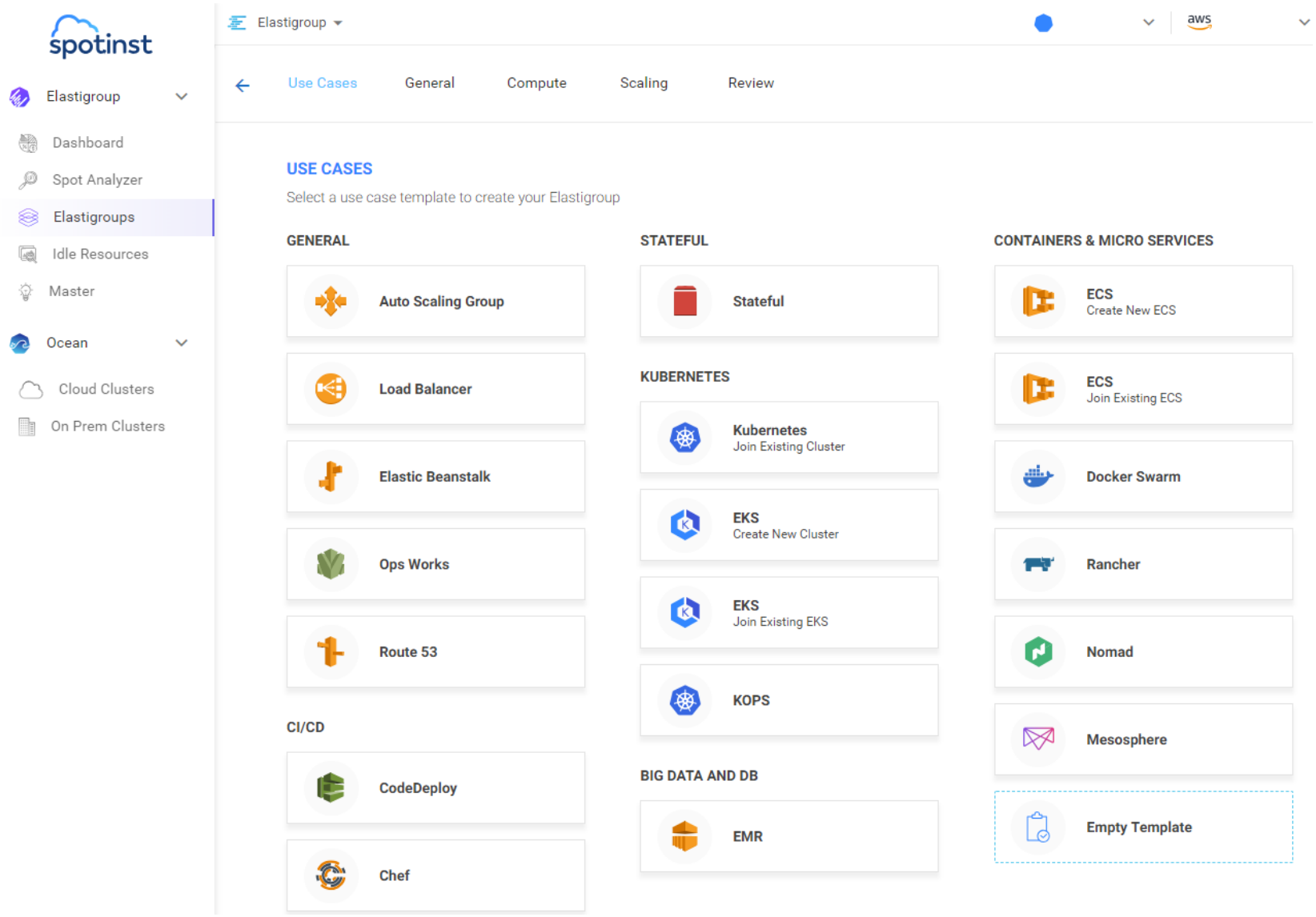
AWS Batch


AWS DATA
FILES
APPS
S3
Send post-trade data to AWS object storage
AWS BATCH
Using AWS Batch, configure resources and schedule when to run data analytics workloads
BIG DATA
Run Big Data analytics tools for appropriate analysis and market predictions for next day trading
STORAGE
Store results for further analysis or long-term

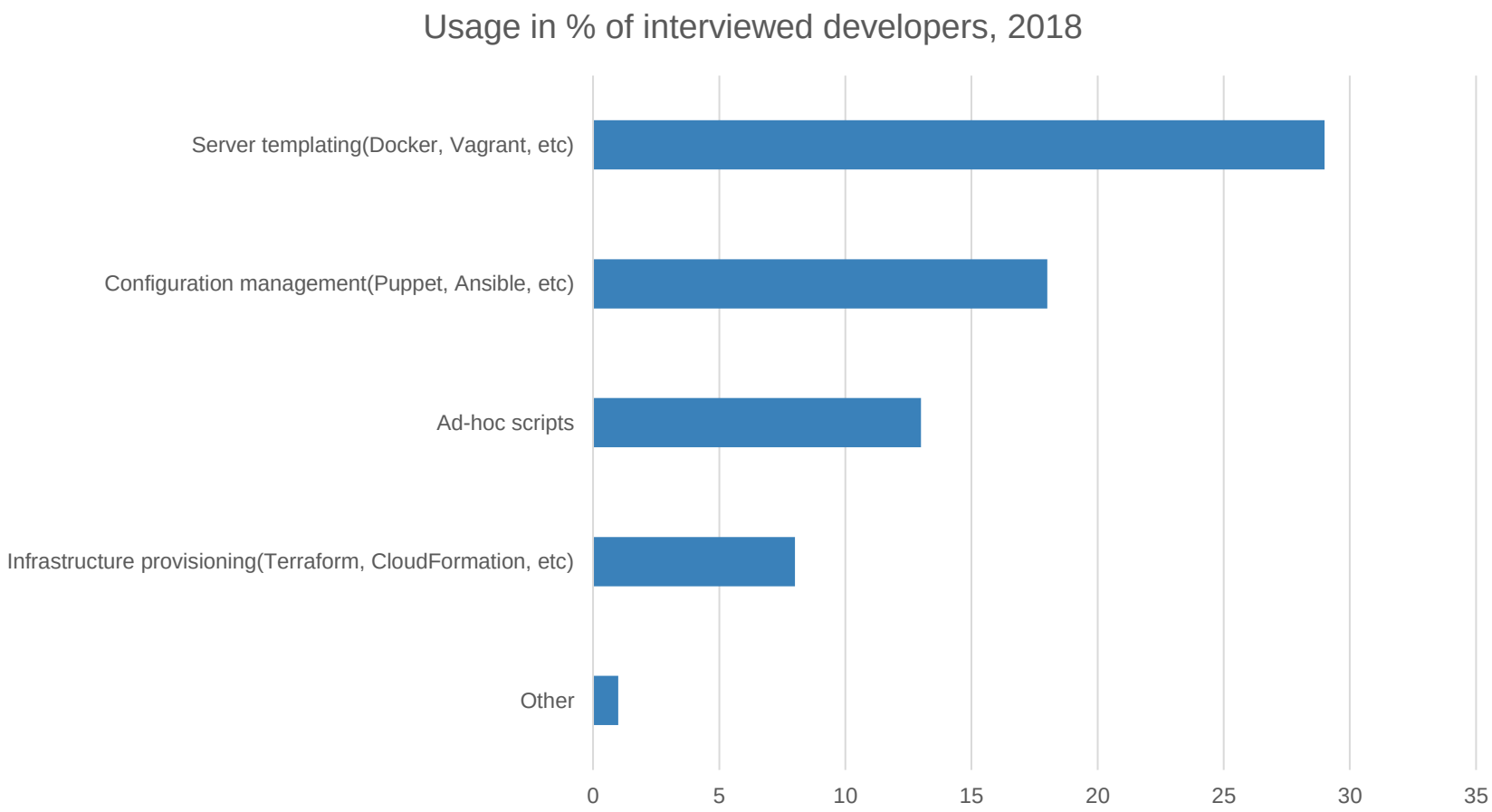
AWS Free Tier



Управление Spots

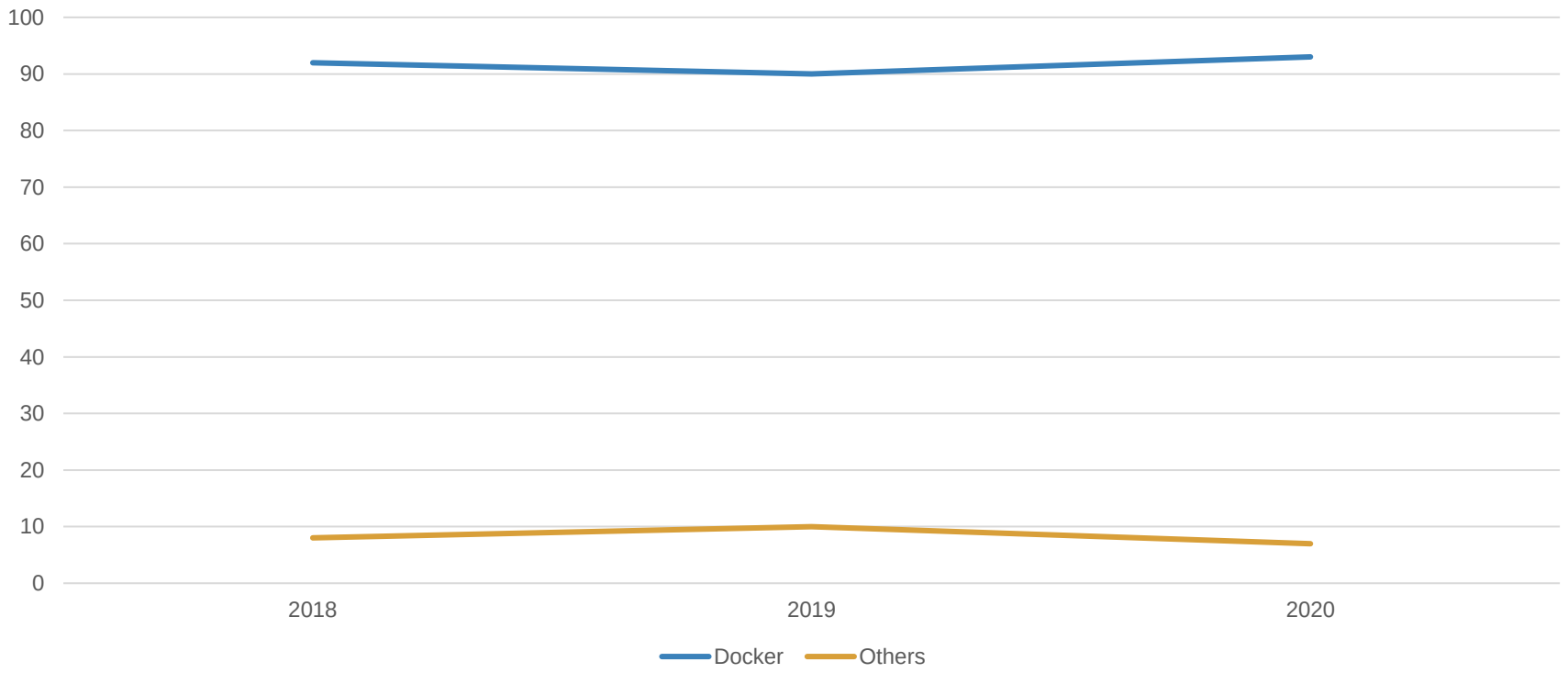


Infrastructure-As-a-Code tools



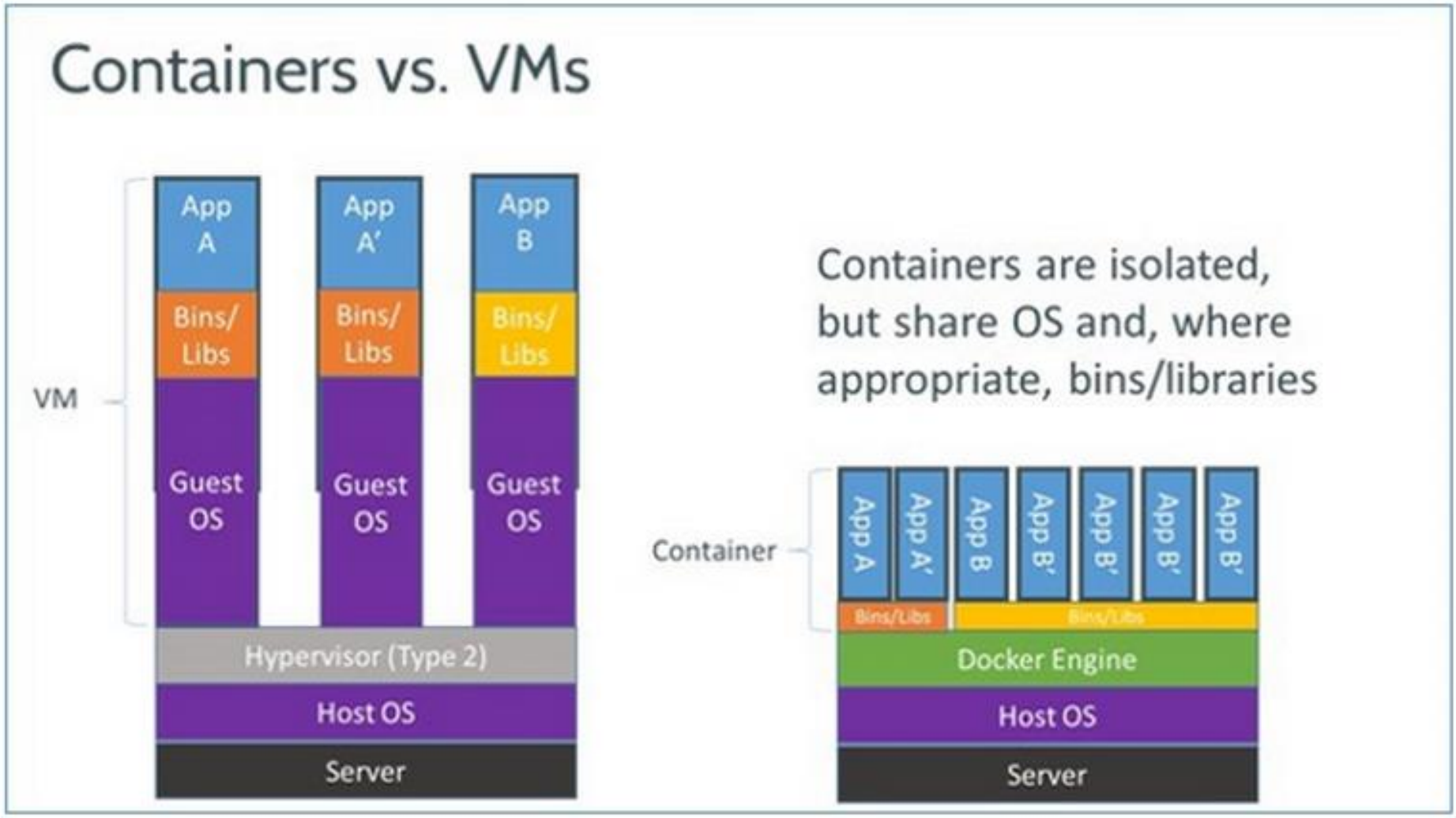
Источник: https://www.jetbrains.com/research/devecosystem-2018/devops/

Server templating tools usage (%)

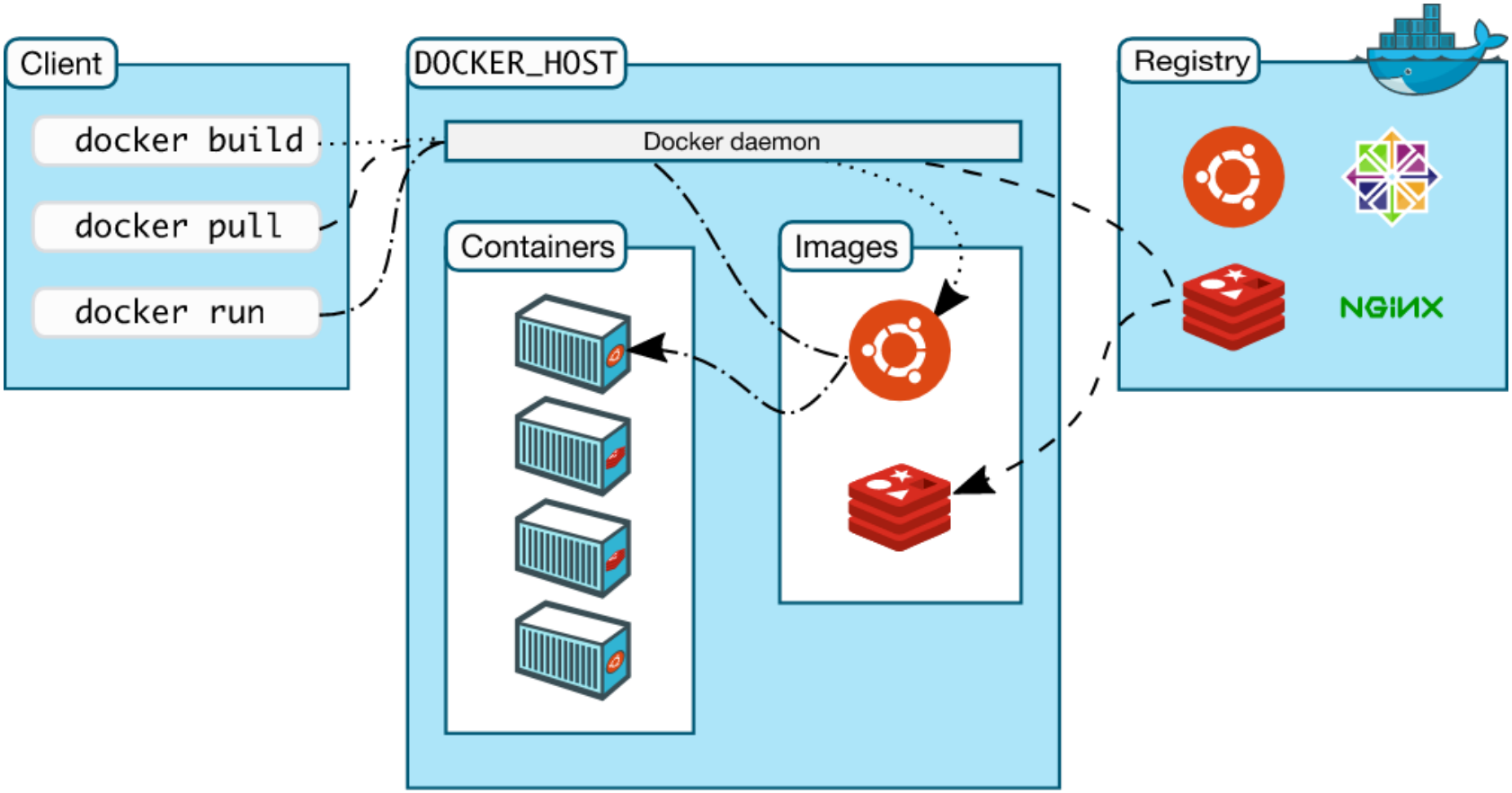


Источник: https://www.jetbrains.com/research/devecosystem-2018/devops/

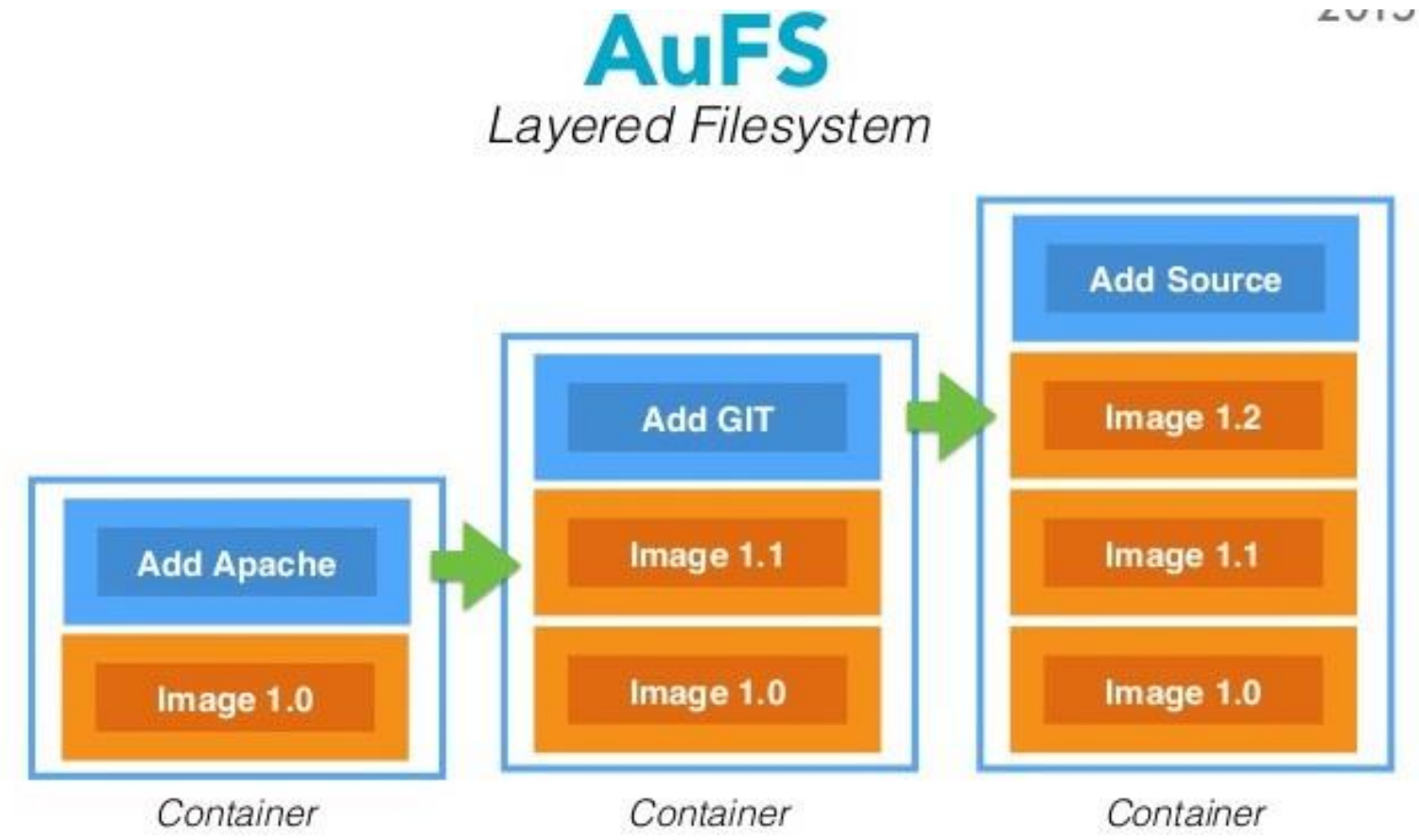
Docker


Containers vs. VMs
VM
App A
App A'
App B
Bins/ Libs
Bins/ Libs
Bins/ Libs
Guest OS
Guest OS
Guest OS
Hypervisor (Type 2)
Host OS
Server
Containers are isolated, but share OS and, where appropriate, bins/libraries
Container
App A
App A'
App B
App B'
App B'
App B'
App B'
Bins/Libs
Bins/Libs
Docker Engine
Host OS
Server

Docker


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

Docker Layers

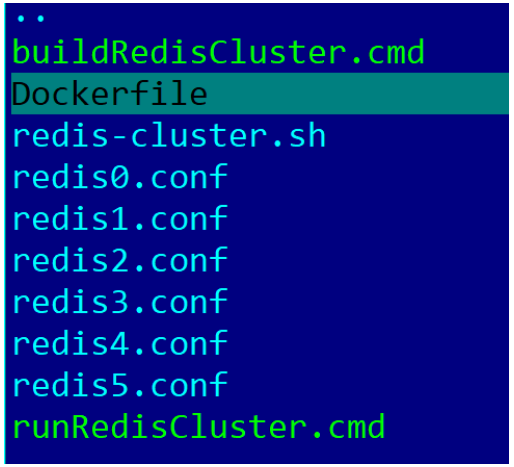


Example: Run Aerospike image with custom version/config/port

```
docker run -d -v C:\Projects\docker-files\aerospike:/opt/aerospike/etc --name aerospike-jarsoft -p 3000:3000 -p 3001:3001 -p 3002:3002 aerospike:4.8.0.2 asd --foreground --config-file /opt/aerospike/etc/aerospike-jarsoft.conf
```

Example: Redis Cluster

```
FROM redis:5.0.6
MAINTAINER aden@jarsoft.ru

COPY redis0.conf /etc/redis0.conf
COPY redis1.conf /etc/redis1.conf
COPY redis2.conf /etc/redis2.conf
COPY redis3.conf /etc/redis3.conf
COPY redis4.conf /etc/redis4.conf
COPY redis5.conf /etc/redis5.conf

RUN mkdir -p /var/log/redis
RUN chmod 0777 /var/log/redis
RUN mkdir -p /var/lib/redis
RUN chmod 0777 /var/lib/redis

COPY redis-cluster.sh /etc/redis-cluster.sh
RUN chmod +x /etc/redis-cluster.sh

CMD /etc/redis-cluster.sh

EXPOSE 7000
EXPOSE 7001
EXPOSE 7002
EXPOSE 7003
EXPOSE 7004
EXPOSE 7005
```



Example: Redis Cluster

```
..
buildRedisCluster.cmd
Dockerfile
redis-cluster.sh
redis0.conf
redis1.conf
redis2.conf
redis3.conf
redis4.conf
redis5.conf
runRedisCluster.cmd
```

```
#!/bin/sh

echo "start cluster"

nohup redis-server /etc/redis0.conf > /var/log/redis/redis0.log &
nohup redis-server /etc/redis1.conf > /var/log/redis/redis1.log &
nohup redis-server /etc/redis2.conf > /var/log/redis/redis2.log &
nohup redis-server /etc/redis3.conf > /var/log/redis/redis3.log &
nohup redis-server /etc/redis4.conf > /var/log/redis/redis4.log &
nohup redis-server /etc/redis5.conf > /var/log/redis/redis5.log &

sleep 3

tail -10 /var/log/redis/redis0.log
tail -10 /var/log/redis/redis1.log
tail -10 /var/log/redis/redis2.log
tail -10 /var/log/redis/redis3.log
tail -10 /var/log/redis/redis4.log
tail -10 /var/log/redis/redis5.log

if [ -f "/etc/redis-inited.txt" ]; then
        echo 'cluster is already inited'
else
                echo 'yes' | redis-cli --cluster create 127.0.0.1:7000 127.0.0.1:7001 127.0.0.1:7002 127.0.0.1:7003
127.0.0.1:7004 127.0.0.1:7005 --cluster-replicas 1
                touch /etc/redis-inited.txt
fi

tail -f /var/log/redis/redis0.log
```

Example: Redis Cluster

redis0.conf

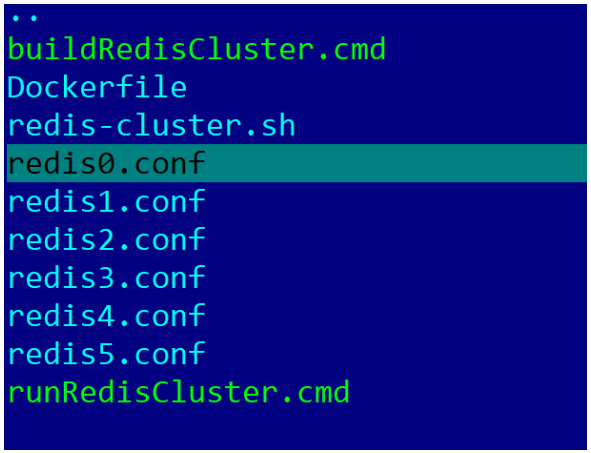
```
port 7000
cluster-enabled yes
cluster-config-file /etc/nodes0.conf
cluster-node-timeout 5000
appendonly yes
```

redis1.conf

```
port 7001
cluster-enabled yes
cluster-config-file nodes1.conf
cluster-node-timeout 5000
appendonly yes
```

redis2.conf

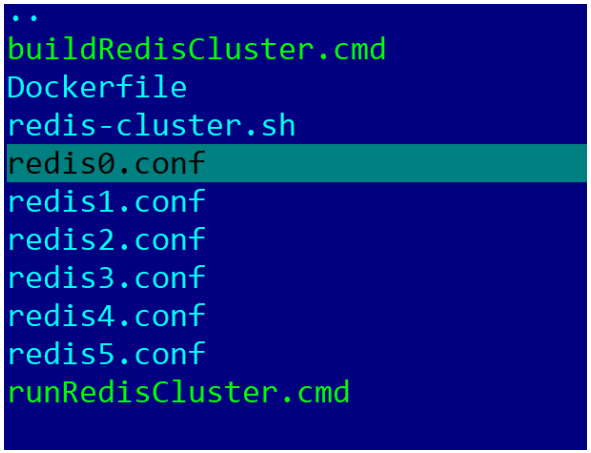
```
port 7002
cluster-enabled yes
cluster-config-file nodes2.conf
cluster-node-timeout 5000
appendonly yes
```



Example: Redis Cluster

buildRedisCluster.cmd

docker build --tag redis-cluster-jarsoft .



buildRedisCluster.cmd

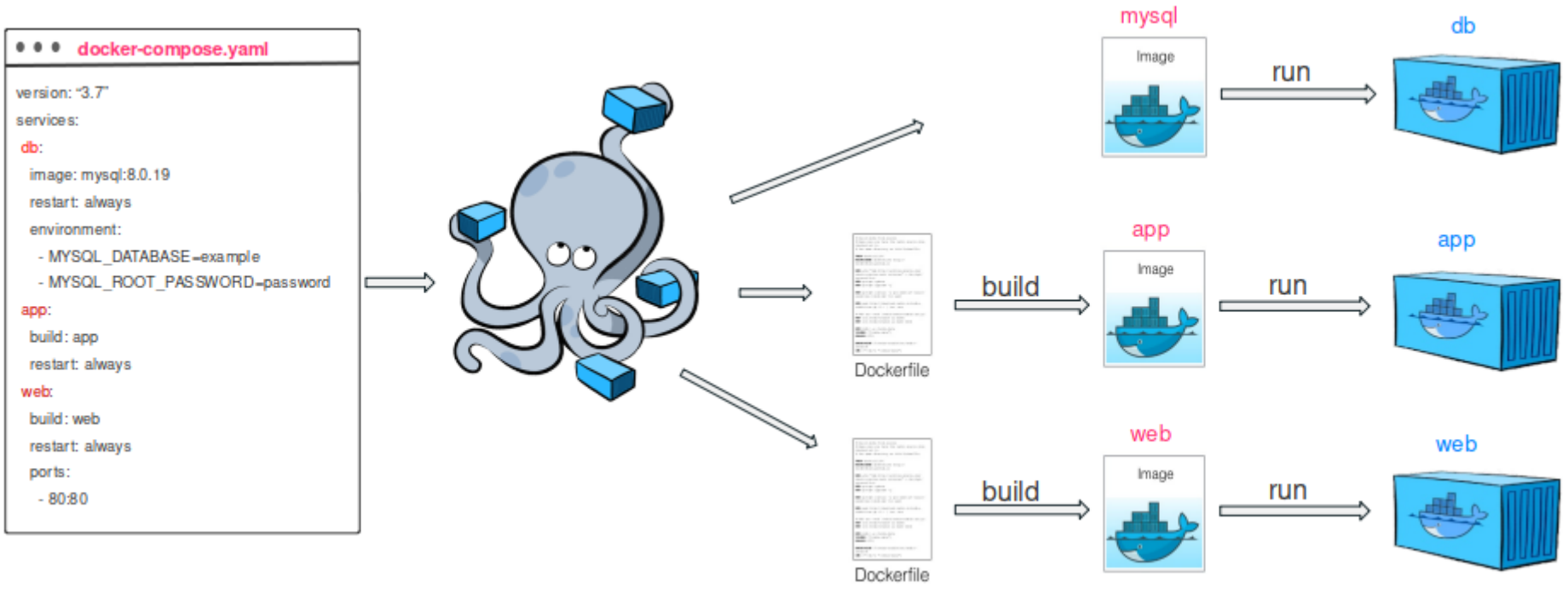
start docker run -it -p 7000:7000 -p 7001:7001 -p 7002:7002 -p 7003:7003 -p 7004:7004 -p 7005:7005 redis-cluster-jarsoft

Example: Druid & Kafka & Registry

```
FROM openjdk:8

ENV DRUID_VERSION 0.19.0

# Get Druid
RUN mkdir -p /tmp \
    && cd /tmp/ \
    && curl -fsLS "http://mirror.linux-ia64.org/apache/druid/0.19.0/apache-druid-${DRUID_VERSION}-bin.tar.gz" | tar xvz \
    && mv apache-druid-$DRUID_VERSION /opt/druid

RUN sed -i 's/druid\.extensions\.loadList.*/druid.extensions.loadList=["druid-hdfs-storage", "druid-kafka-indexing-service", "druid-datasketches", "druid-avro-extensions", "druid-parquet-extensions", "druid-kafka-indexing-service"]]/' /opt/druid/conf/druid/single-server/nano-quickstart/_common/common.runtime.properties

RUN sed -i 's/druid\.worker\.capacity.*/druid\.worker\.capacity=10/' /opt/druid/conf/druid/single-server/nano-quickstart/middleManager/runtime.properties

RUN mkdir -p /tmp \
    && cd /tmp/ \
    && curl -fsLS "http://packages.confluent.io/archive/5.5/confluent-5.5.1-2.12.tar.gz" | tar xvz \
    && mv confluent-5.5.1 /opt/confluent

RUN sed -i 's/listeners=.*/listeners=http:\/\/0.0.0.0:9081/' /opt/confluent/etc/schema-registry/schema-registry.properties
```

```
RUN sed -i 's/#advertised\.listeners.*/advertised\.listeners=PLAINTEXT:\/\/localhost:9092/'
/opt/confluent/etc/kafka/server.properties

RUN echo "auto.create.topics.enable=true" >> /opt/confluent/etc/kafka/server.properties

COPY druid-kafka.sh /opt/druid-kafka.sh
RUN chmod 0755 /opt/druid-kafka.sh

# Expose ports:
# - 8888: Druid router
# - 8081: Druid coordinator/overlord
# - 8082: Druid broker
# - 8083: Druid historical
# - 2181 2888 3888: ZooKeeper
# - 8090: Kafka supervisor
# - 9092: Kafka
# - 9081: Schema-registry (moved from default 8081)

EXPOSE 8888
EXPOSE 8081
EXPOSE 8082
EXPOSE 8083
EXPOSE 2181 2888 3888
EXPOSE 8090
EXPOSE 9092
EXPOSE 9081

ENTRYPOINT /opt/druid-kafka.sh
```

Docker Compose



Docker Compose

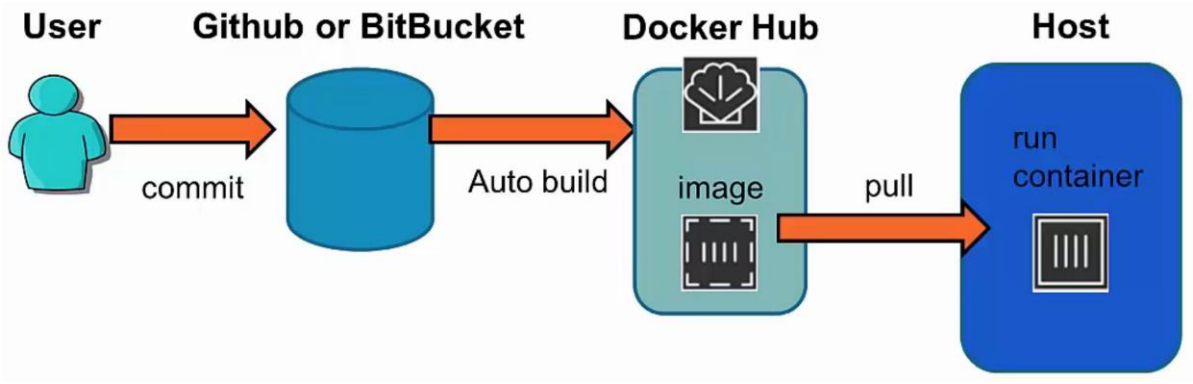
/usr/local/bin/docker-compose up --abort-on-container-exit --quiet-pull

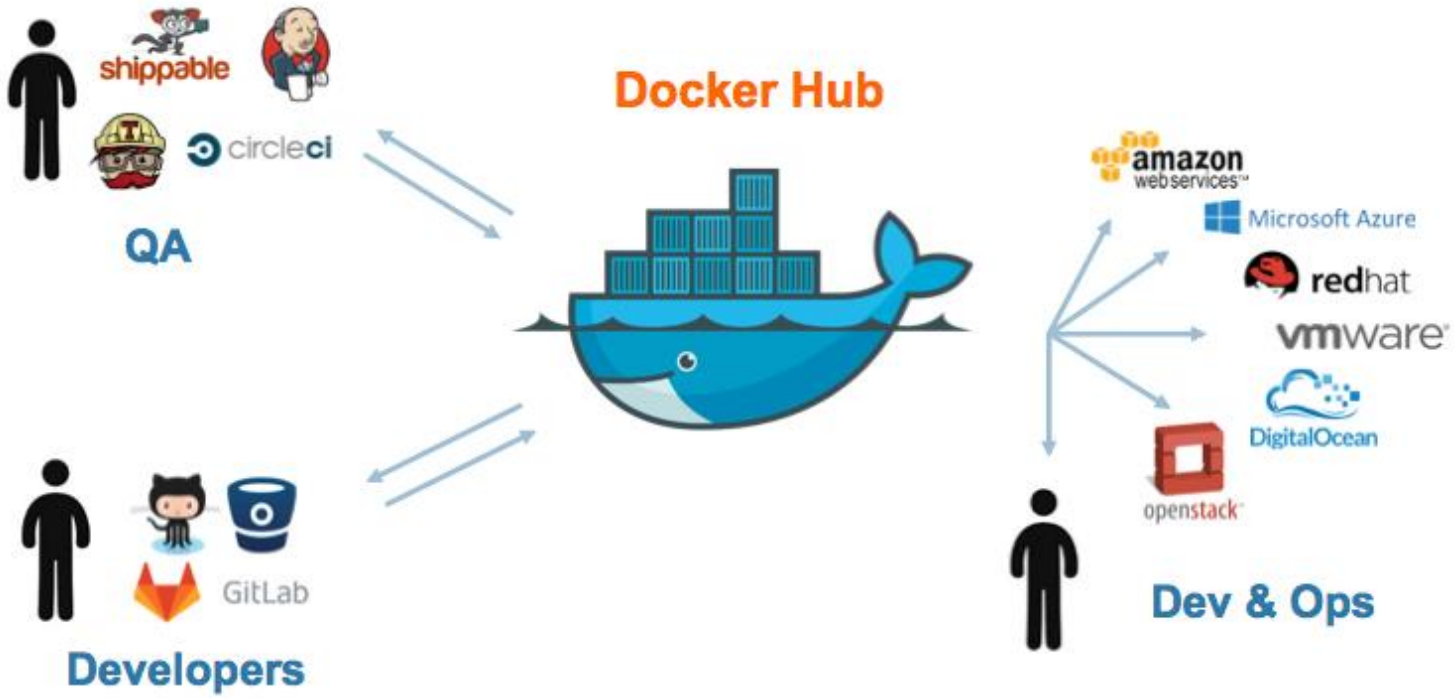
docker-compose.yaml

```
version: '2.1'
services:
 config:
   container_name: config
   image: registry.jarsoft.ru/qa/config:qa-latest
   ports:
     - "8010:8010"
   healthcheck:
     test: "curl -f http://config:8010/ui/qa"

 mysql:
   container_name: mysql
   image: registry.jarsoft.ru/qa/mysql:5.7
   ports:
     - "3306:3306"
   volumes:
     - "/data/mysql/data:/var/lib/mysql"
   healthcheck:
     test: "/usr/bin/mysql --user=root --password=root --execute \"USE atomik;\""
```

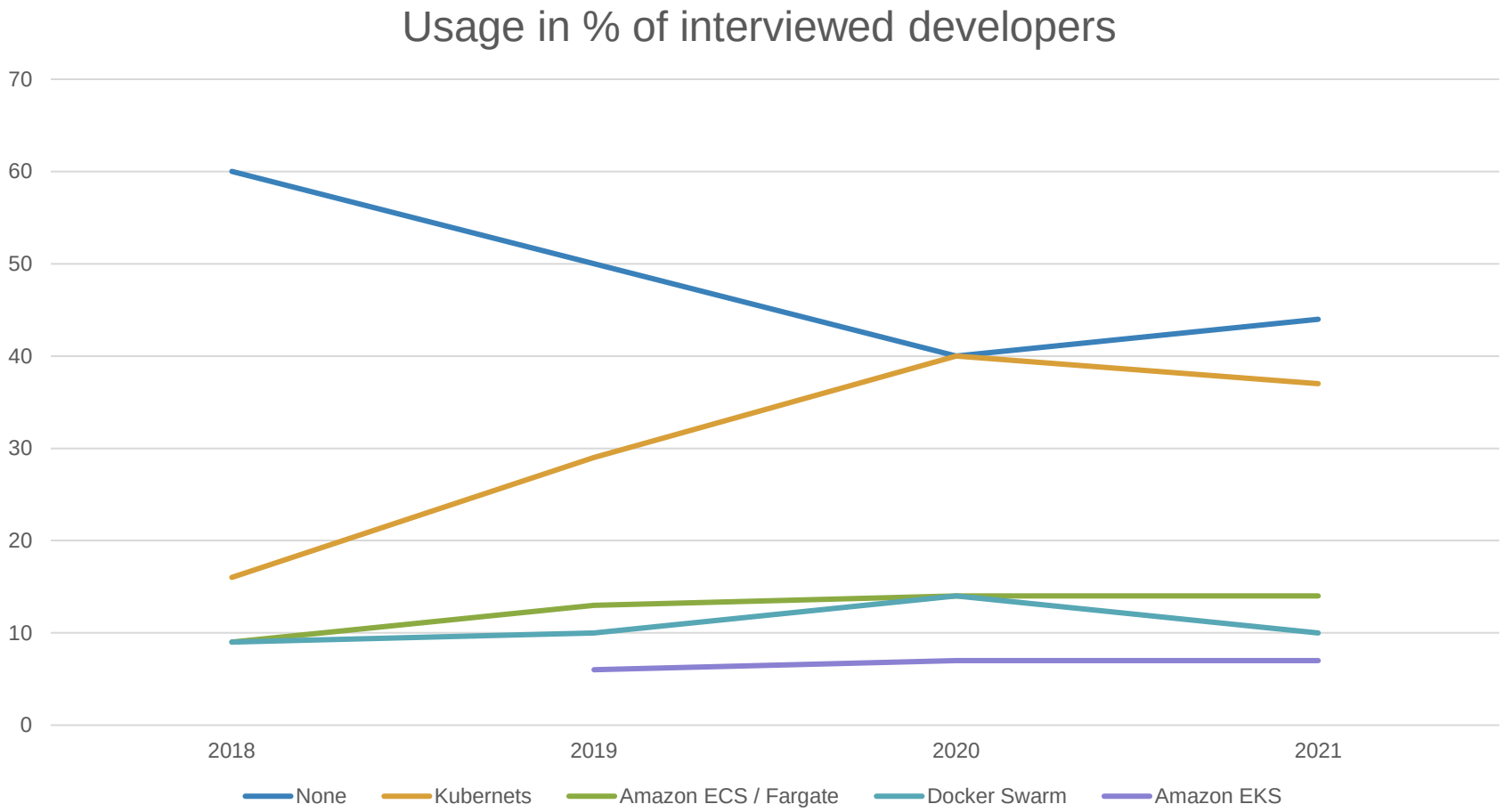
```
aerospike:
  container_name: aerospike
  image: aerospike:4.3.0.7
  logging:
    driver: none
  ports:
    - "3000:3000"
  command: ["asd", "--foreground", "--config-file", "/opt/aerospike/etc/aerospike.conf"]
  volumes:
    - "/data/aerospike/data:/opt/aerospike/data"
    - "/data/aerospike/etc:/opt/aerospike/etc"
    - "/data/aerospike/log:/var/log/aerospike"
  healthcheck:
    test: ["CMD", "asinfo", "-v", "status"]

ui:
  container_name: ui
  image: registry.jarsoft.ru/qa/ui:qa-latest
  ports:
    - "8080:8080"
  healthcheck:
    test: "curl -f http://ui:8080/build-info"
    retries: 10
  depends_on:
    mysql:
      condition: service_healthy
    aerospike:
      condition: service_healthy
    config:
      condition: service_healthy
```

Docker Hub


User
Github or BitBucket
Docker Hub
Host
commit
Auto build
image
pull
run
container

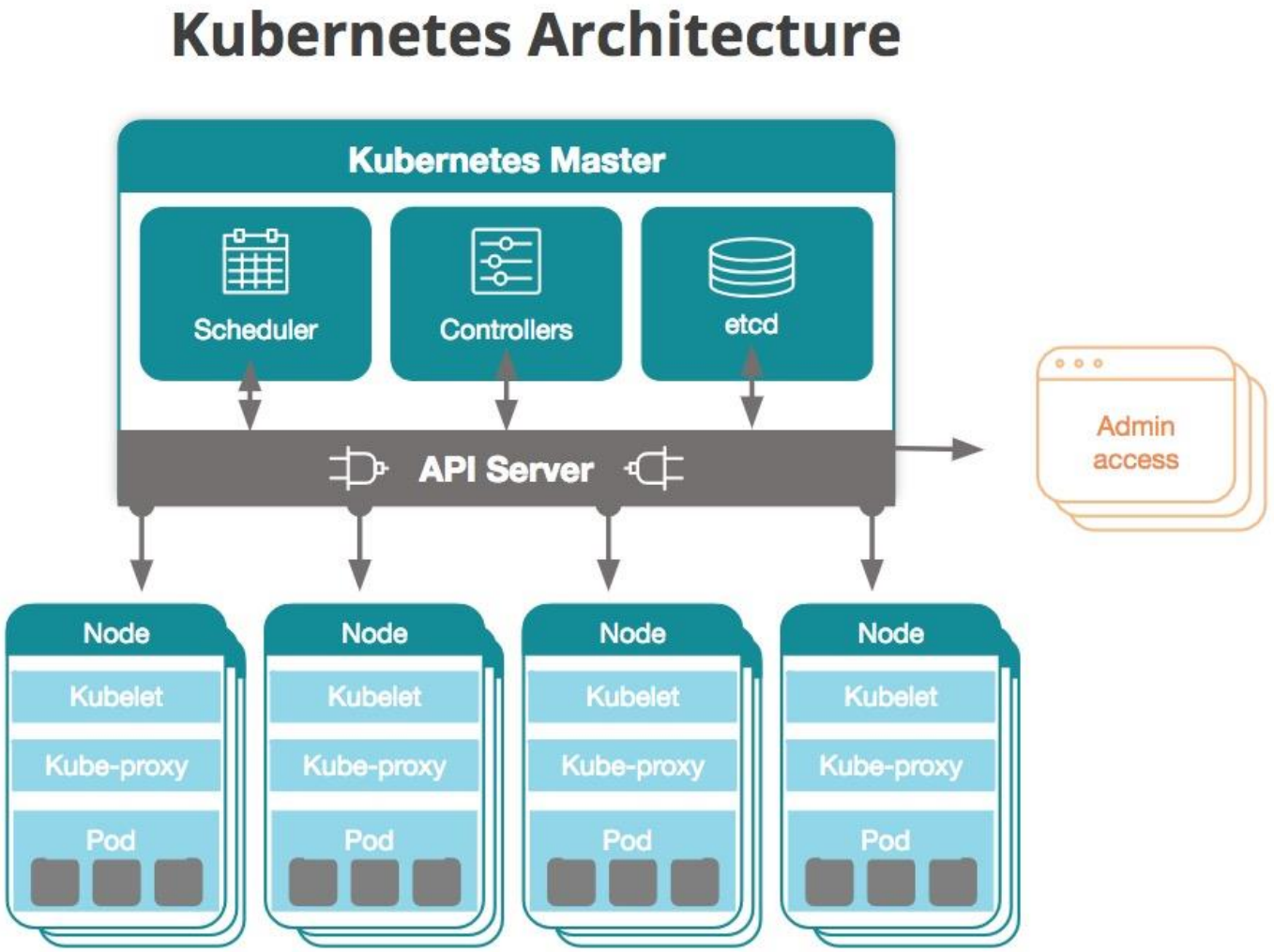

shippable
circleci
QA
Docker Hub
amazon
web services
Microsoft Azure
redhat
vmware
DigitalOcean
openstack
GitLab
Developers
Dev & Ops

Container orchestration services

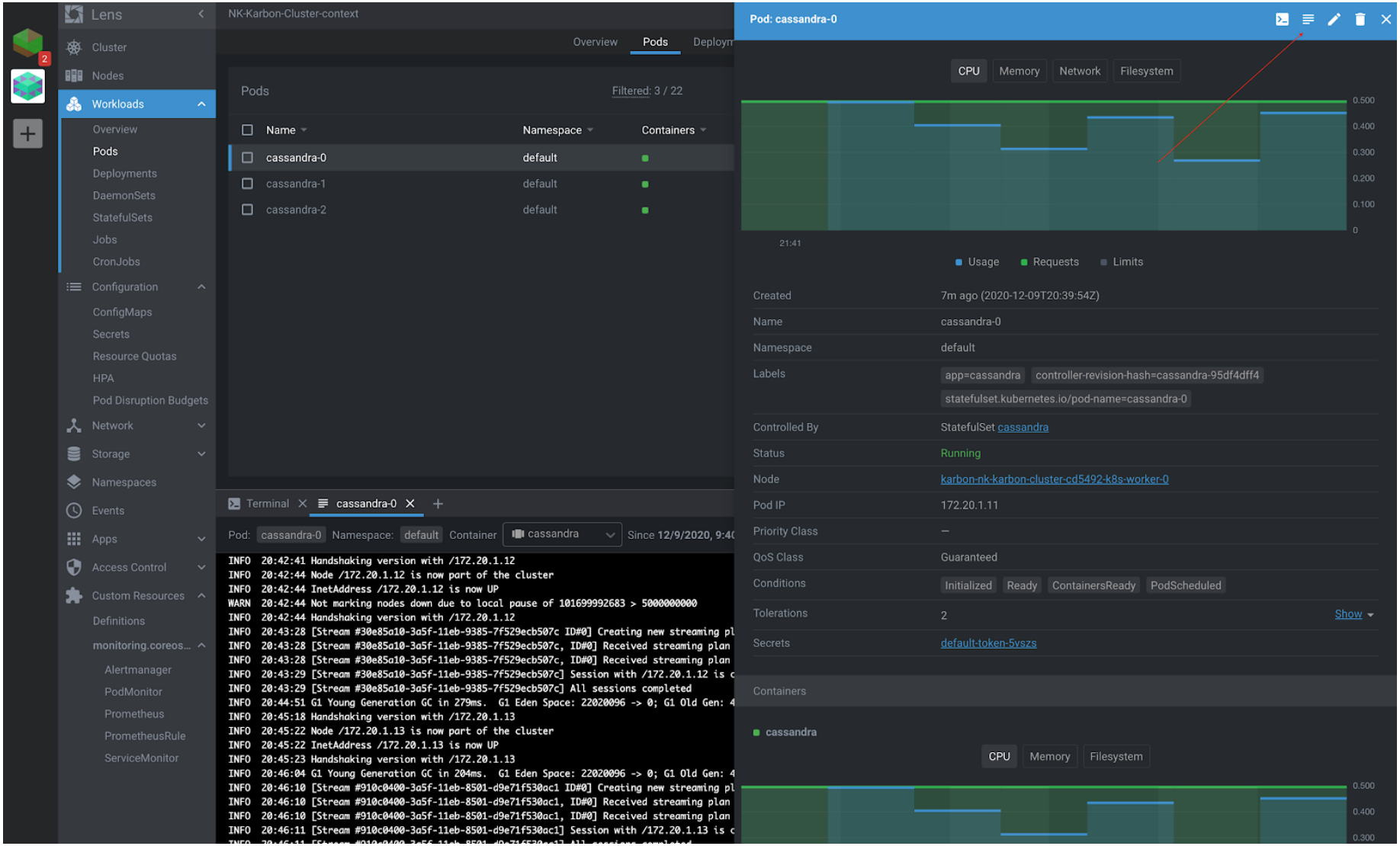


Источник: https://www.jetbrains.com/lp/devecosystem-2021/devops/

Kubernetes


Kubernetes Architecture
Kubernetes Master
Scheduler
Controllers
etcd
API Server
Admin access
Node
Kubelet
Kube-proxy
Pod
Node
Kubelet
Kube-proxy
Pod
Node
Kubelet
Kube-proxy
Pod
Node
Kubelet
Kube-proxy
Pod

Kubernets IDE



Рекомендуемая литература

- http://www.opscode.com/chef/
- http://saltstack.com/
- http://ansible.cc
- http://www.cfengine.com/
- http://www.puppetlabs.com/
- http://vagrantup.com/