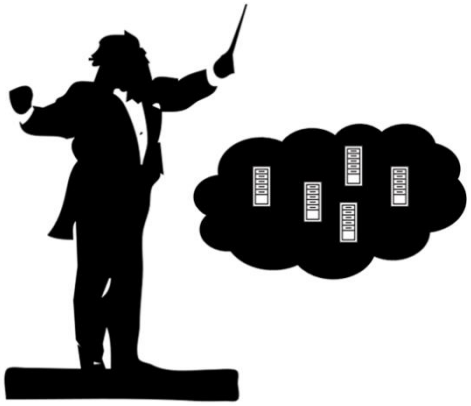
КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ

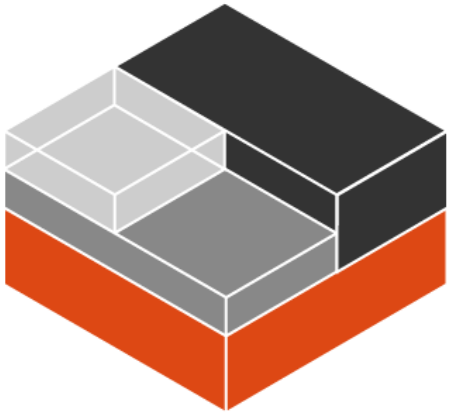
Author:

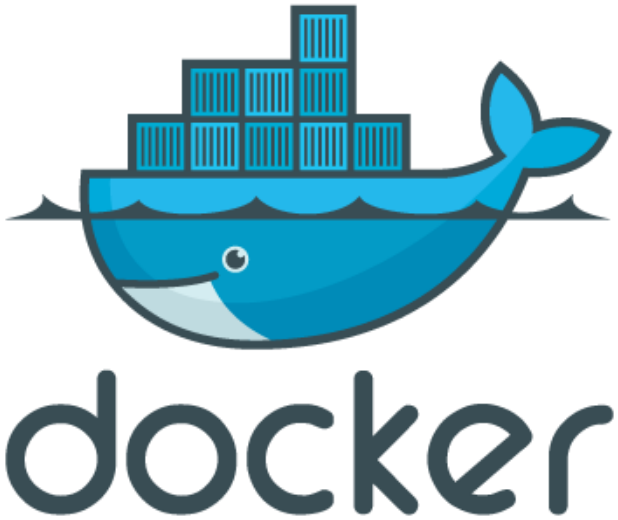
Contents



INTRODUCTION TO CONTAINER ORCHESTRATION

От LXC до Docker

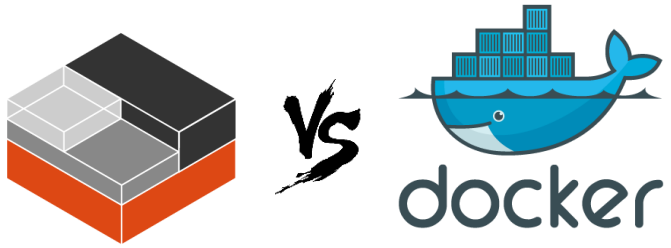




Introduction to container orchestration

LXC VS Docker

1. Количество процессов
2. Структурность
3. Портативность

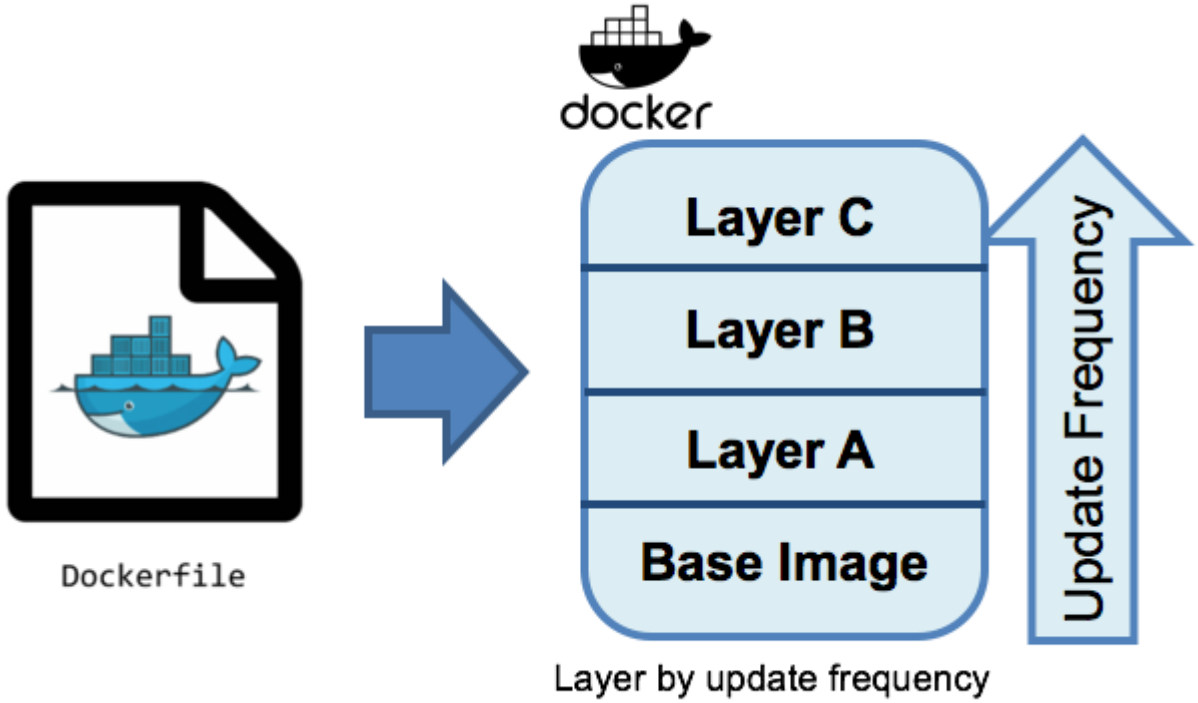


Контейнер

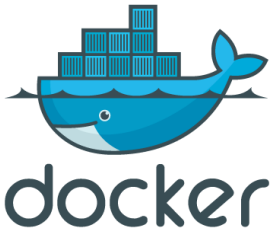
Виртуальные машины

docker

Образ контейнера Docker



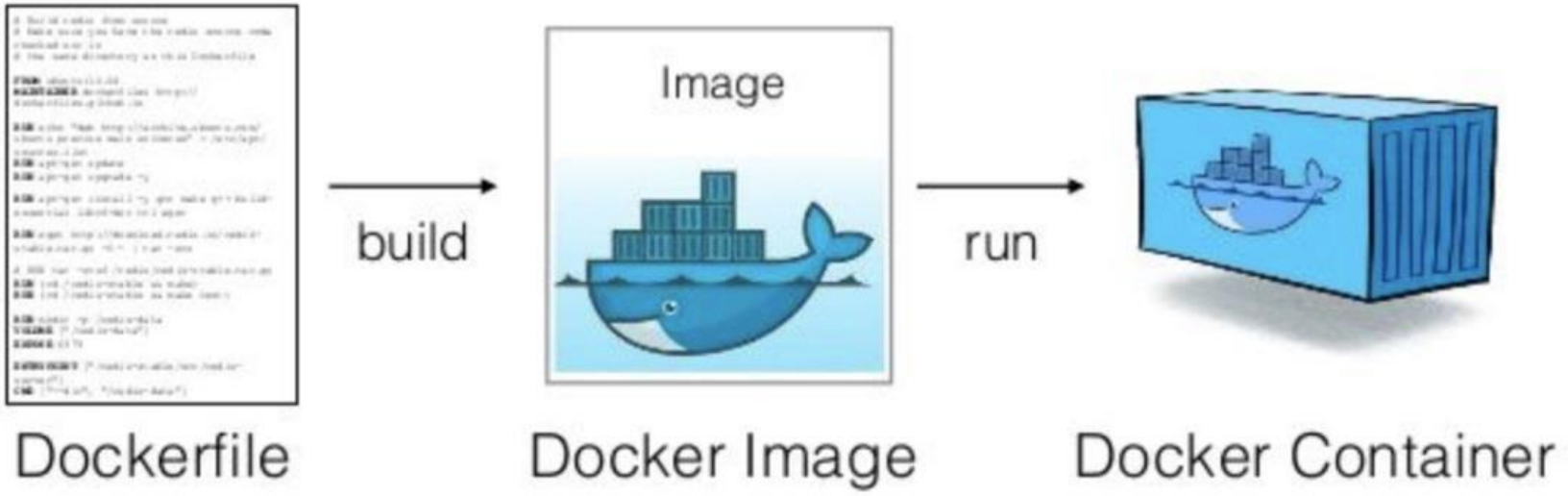
Dockerfile



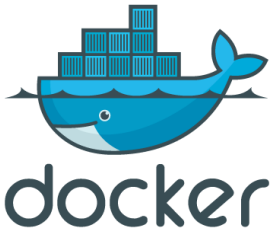

```
FROM centos:7
MAINTAINER Ivan Govorukhin
RUN yum update
RUN yum install -y nginx
RUN echo 'Hi, I am in your container' \
        >/usr/share/nginx/html/index.html
ENV REFREASHED_AT 20190830
EXPOSE 80
WORKDIR /opt/webapp/db
USER user
VOLUME ["/opt/project"]
ADD software.lic
/opt/application/software.lic
COPY conf.d/ /etc/httpd/
ONBUILD ADD . /app/src

ENTRYPOINT ["/usr/sbin/nginx"]
CMD ["-g","daemon","off"]
```

Dockerfile



Dockerfile



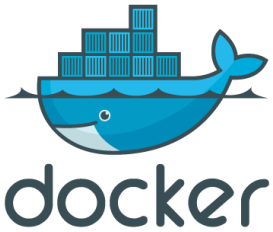
Сборка образа:

```
docker build –t mynginx .
```

Запуск образа

```
docker run --name nginx mynginx
```

Dockerfile



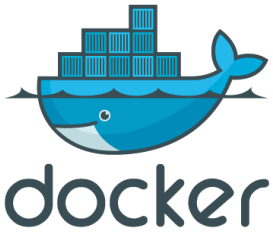
Выставление порта

```
docker run –p 80:80 --name nginx mynginx
```

Сохранение данных

```
docker run –v /path/to/folder:/docker/folder –p 80:80
--name nginx mynginx
```

Dockerfile

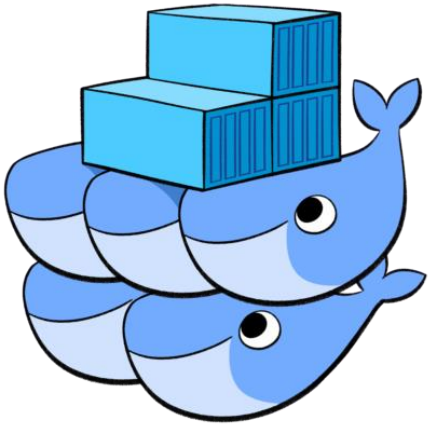


Создание сети

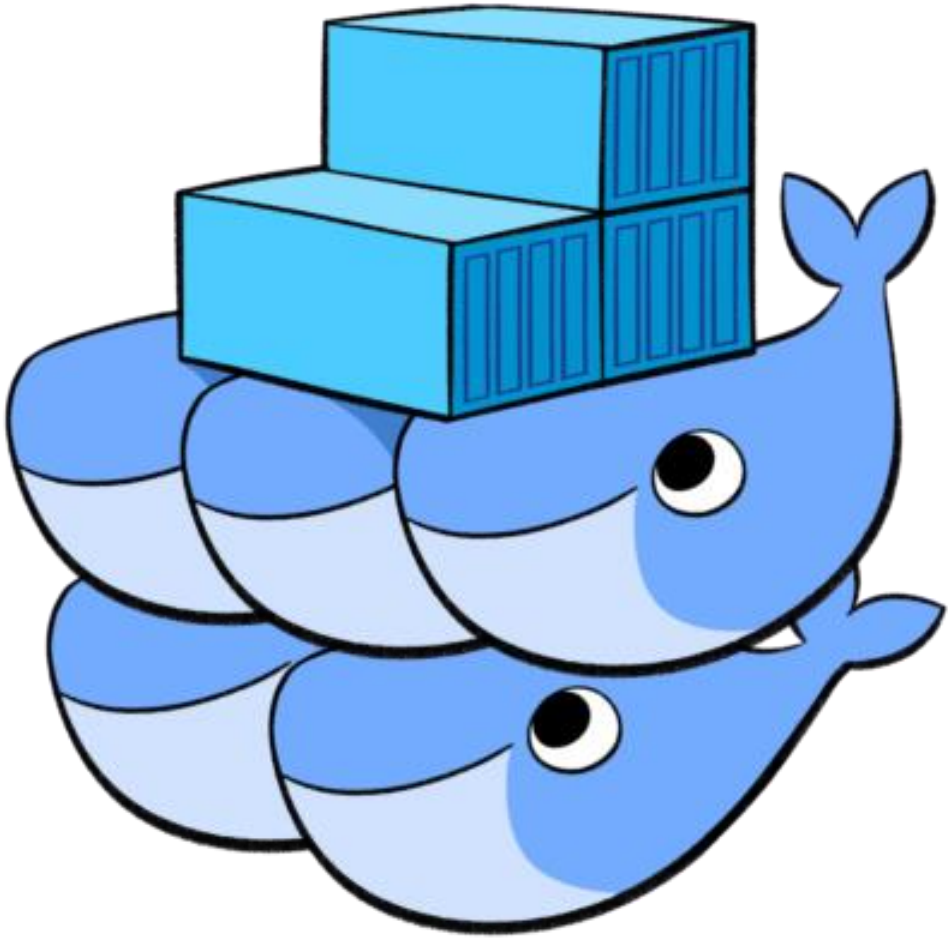
```
docker network create --driver=bridge --subnet=172.18.0.0/16
--gateway 172.18.0.1 epm
```

Использование созданной сети

```
docker run –net epm –p 80:80 --name nginx mynginx
```

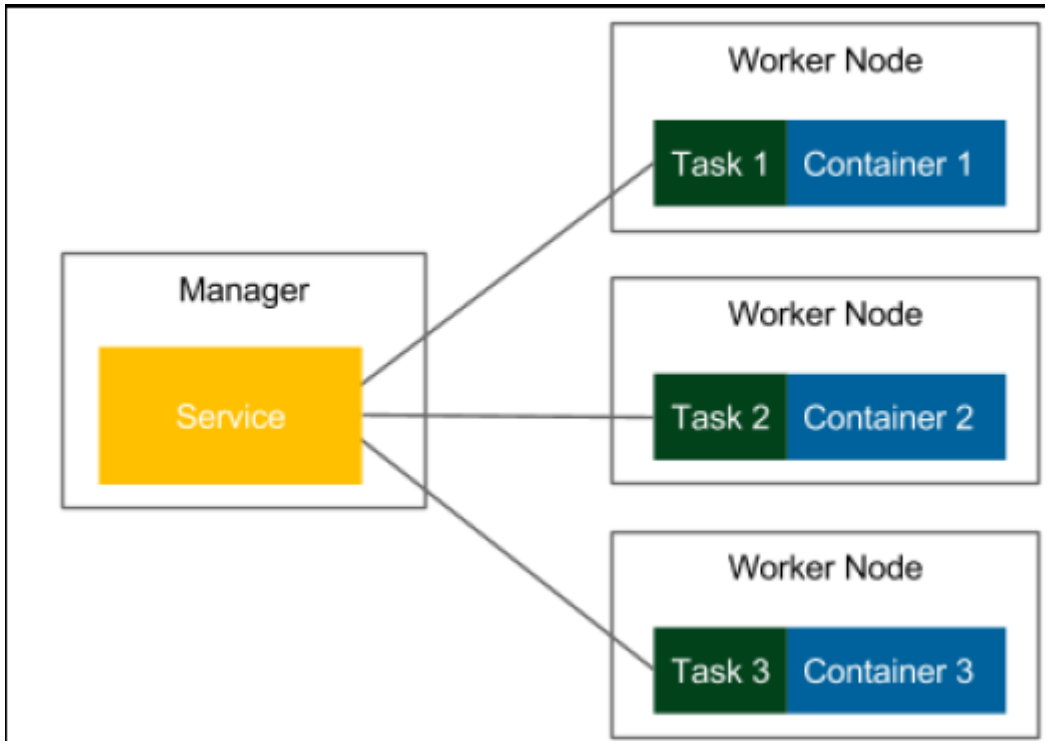


DOCKER SWARM OVERVIEW



Docker Swarm overview

Архитектура Docker Swarm



Возможности Docker Swarm

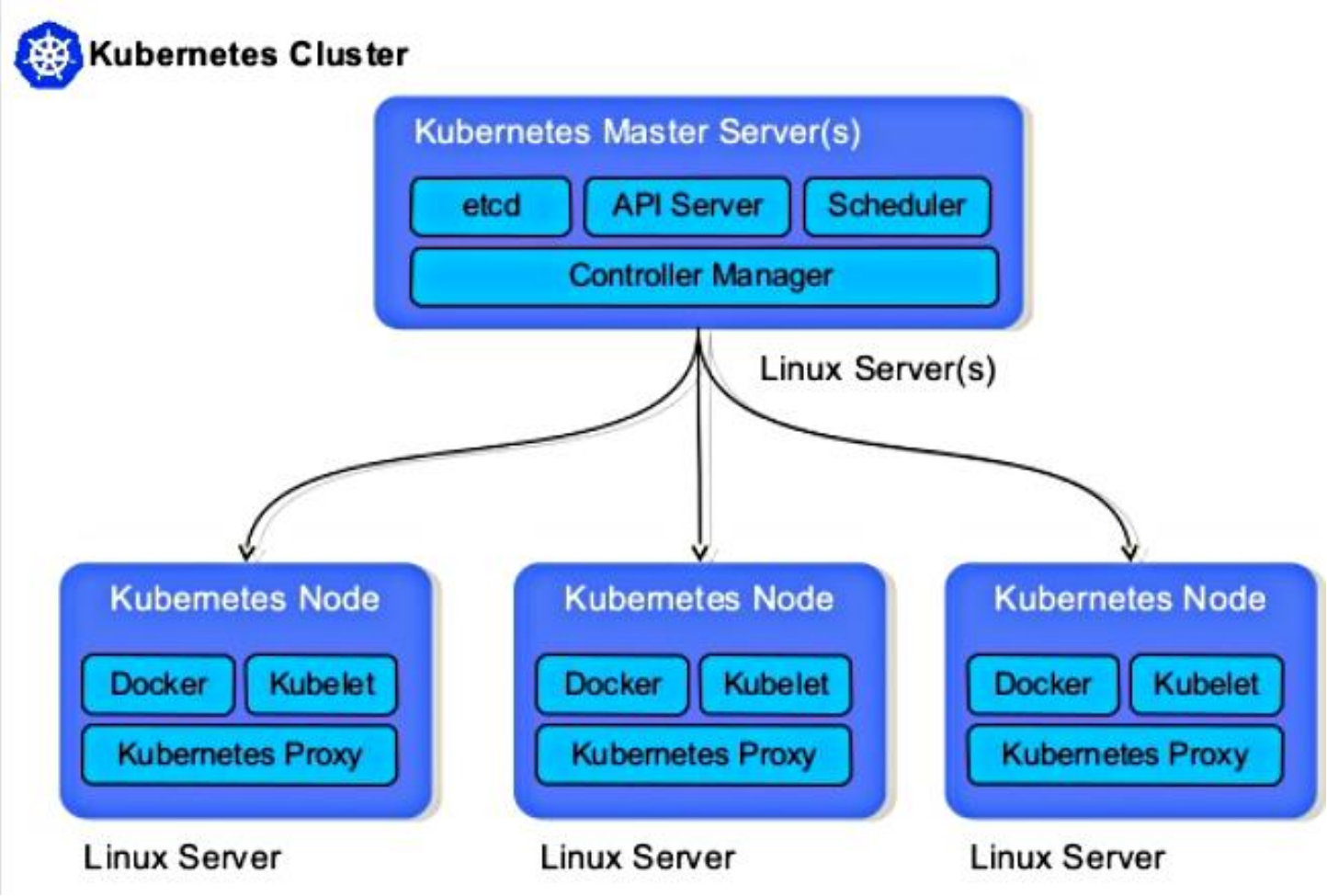
1. Балансировка нагрузки
2. Динамическое управление ролями
3. Динамическое масштабирование сервисов
4. Восстановление отказа
5. Rolling-обновления

KUBERNETES OVERVIEW

Возможности Kubernetes

1. Запуск контейнеров на нодах
2. service discovery, балансировка нагрузки и autoscaling
3. Высокая доступность сервисов

Архитектура Kubernetes



Namespaces

- виртуальные кластера, поддерживаемые одним и тем же физическим кластером
- предназначен для использования в средах с большим количеством пользователей, распределенных по нескольким командам или проектам
- имена ресурсов должны быть уникальными в namespace, но не в разных namespace’ах.
- namespaces это способ разделения ресурсов кластера между несколькими пользователями (через квоту ресурсов).

Pod

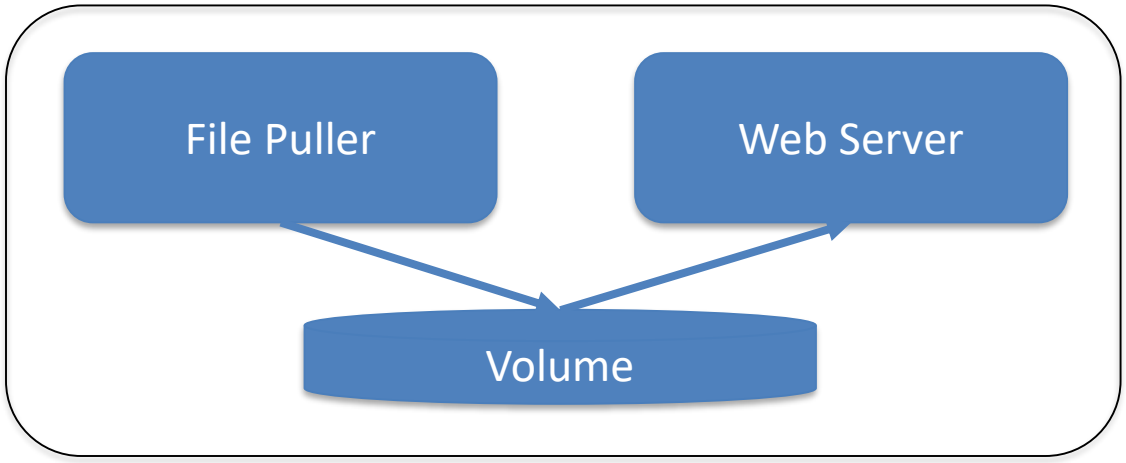
- Является базовым строительным блоком Kubernetes - самым маленьким и простым элементом в объектной модели Kubernetes, которую вы создаете или развертываете.
- Включает в себя контейнер приложения или несколько контейнеров
- Имеет уникальный сетевой IP-адрес в кластере

Multiply containers

- Pod - предназначены для поддержки нескольких взаимодействующих процессов (в виде контейнеров)
- Контейнеры в pod’е автоматически размещаются и запускаются на одной физической или виртуальной машине в кластере.
- Контейнеры могут совместно использовать ресурсы и зависимости, общаться друг с другом.
- Pod’ы предоставляют два вида общих ресурсов для составляющих их контейнеров: сеть и хранилище

Multiply containers

- Pod - предназначены для поддержки нескольких взаимодействующих процессов (в виде контейнеров)
- Контейнеры в pod’е автоматически размещаются и запускаются на одной физической или виртуальной машине в кластере.
- Контейнеры могут совместно использовать ресурсы и зависимости, общаться друг с другом.
- Pod’ы предоставляют два вида общих ресурсов для составляющих их контейнеров: сеть и хранилище



Init container

- Являются специализированными контейнерами, которые запускаются перед контейнерами приложения и могут содержать утилиты или сценарии установки, отсутствующие в образе приложения.

ПРЕИМУЩЕСТВА:

- Они могут содержать и запускать утилиты, которые нежелательно включать в образ контейнера приложения из соображений безопасности.
- Нет необходимости создавать образ из другого образа только для того, чтобы использовать такой инструмент, как sed, awk, python или dig во время установки.
- Они запускаются до завершения контейнера с приложений, тогда как контейнеры приложений работают параллельно.
- init контейнеры предоставляют простой способ блокировать или задерживать запуск контейнеров приложений до тех пор, пока не будет выполнен некоторый набор предварительных условий.

Deployment

- Предоставляет обновление для Pod’ов и ReplicaSet
- Изменение фактическоого состояния до желаемого состояния
- Вы не должны управлять ReplicaSets, принадлежащими Deployment

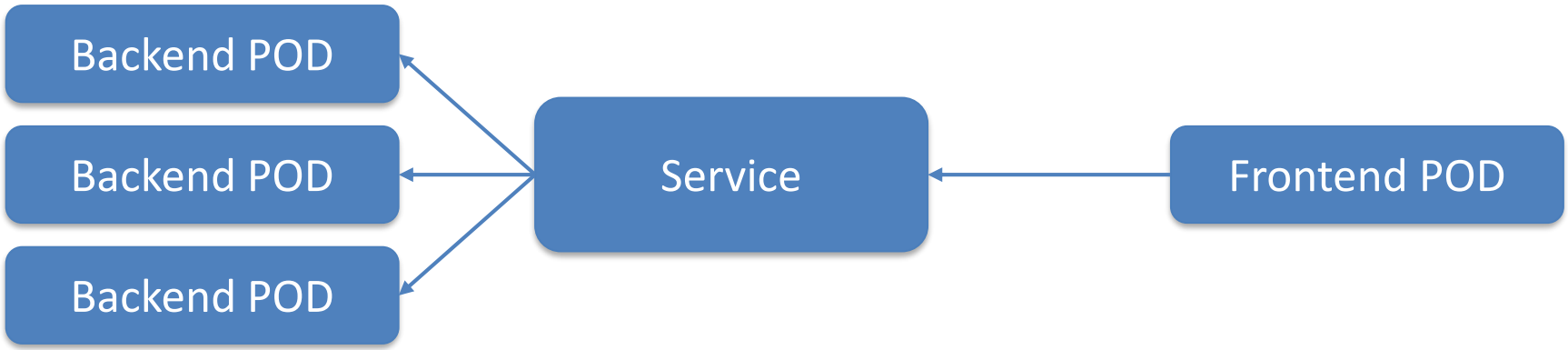
Creating a Deployment

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
```

Services

1. это абстракция, которая определяет логический набор модулей и политику доступа к ним

- Набор Pod’ов, на которые ориентирована Service’ы, определяется Selector’ами.
- “Обычные” (не headless) сервисы вида **my-svc.my-namespace.svc.cluster.local**. Возвращает cluster IP сервиса
- “Headless” (без cluster IP) сервис my-svc.my-namespace.svc.cluster.local возвращает набор ip адресов pod’ов, выбранных сервисом.



Services

```yaml
kind: Service
apiVersion: v1
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Services

ClusterIP

- Выставляет сервис на внутренний ip адрес кластера
- Делает доступным сервис только внутри кластера
- Тип сервиса по умолчанию

NodePort

- Выставляет сервис на каждой ноде на статичный порт
- У вас появляется возможность обращаться к сервису снаружи
- <NodeIP>:<NodePort>

LoadBalancer

- Выставляет сервис используя load balancer облачного провайдера

Переменные окружения

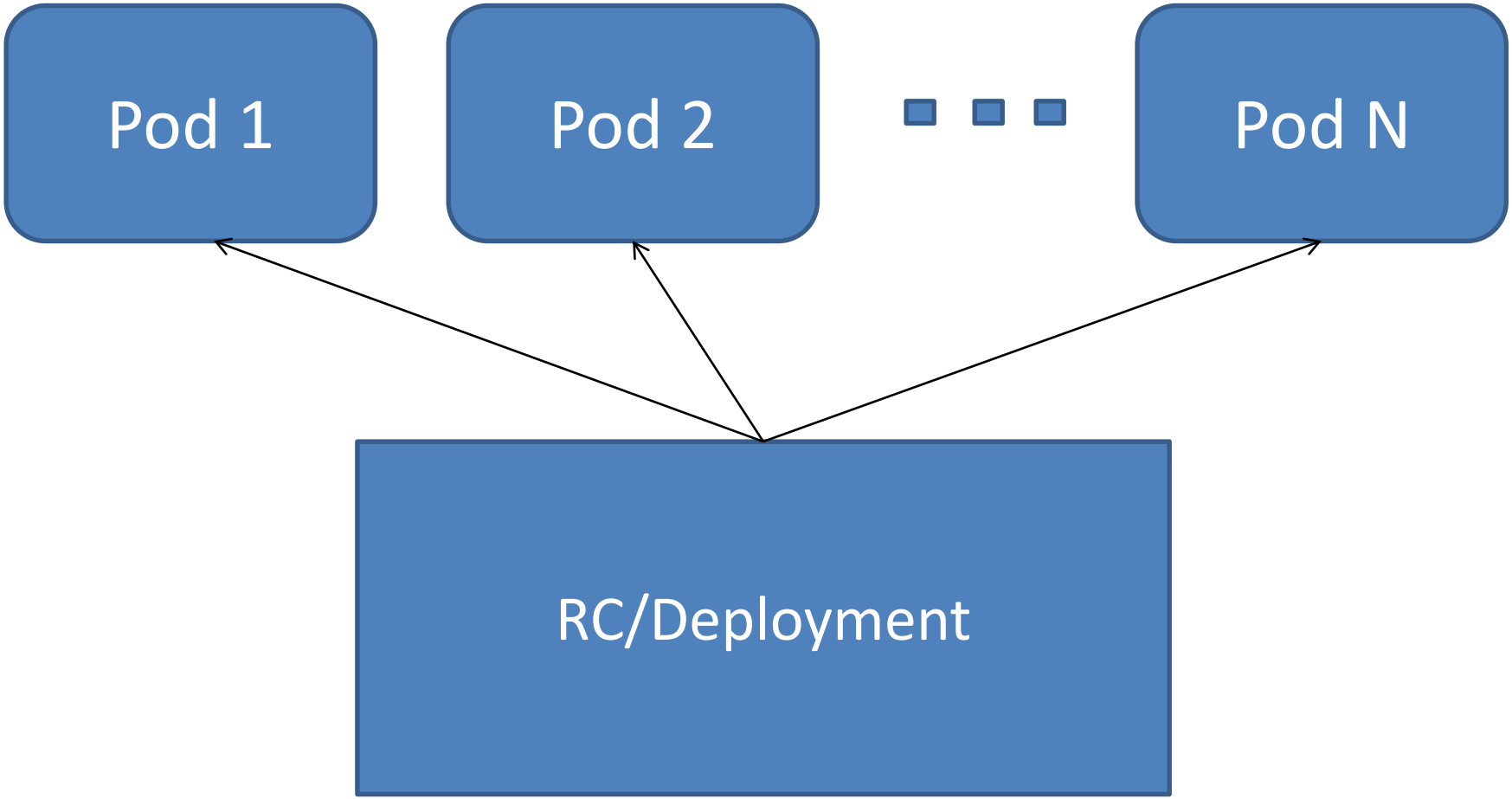
Когда вы создаете Pod, вы можете установить переменные окружения для контейнеров внутри пода.

```
spec:
  containers:
  - name: nginx
    image: nginx:1.12
    ports:
    - containerPort: 80
    env:
    - name: DEMO_KUBERNETES
      value: "Hello kubernetes training"
```

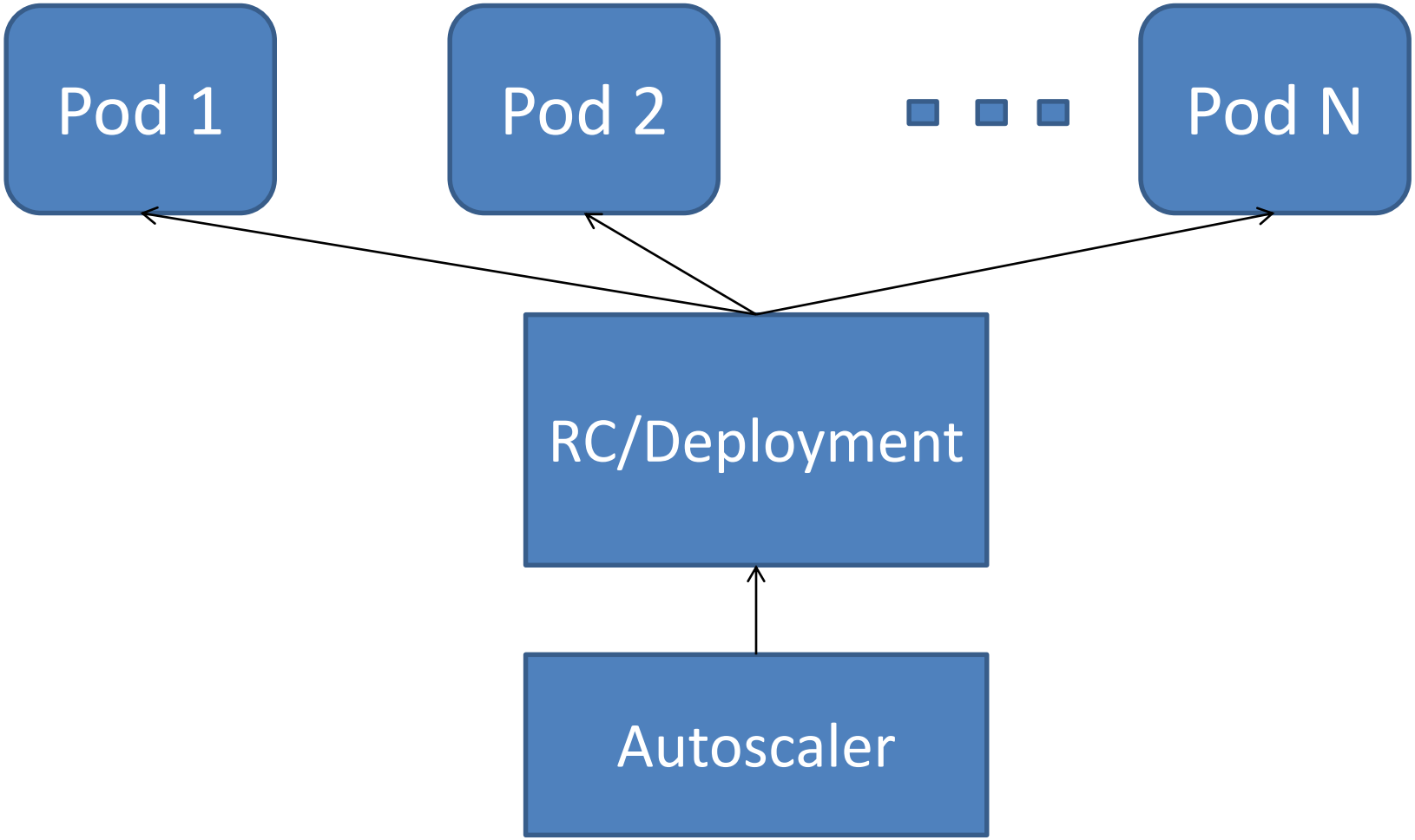
Логи приложения

- Kubectl logs возвращает логи pod’a
- Если в поде размещено несколько контейнеров, необходимо указывать имя контейнера

Application scaling – replica sets



Application scaling – replica sets



Обновление приложения

- KUBECTL SET IMAGE
  - kubectl run nginx --image=nginx:1.12 --replicas=3
  - kubectl rollout status deploy nginx
  - kubectl set image deploy nginx nginx=nginx:1.13
- KUBECTL EDIT
  - kubectl edit deploy nginx
  - kubectl rollout status deploy nginx
- KUBECTL APPLY
  - kubectl get deploy nginx -o yaml > nginx_deployment.yaml
  - kubectl apply -f nginx_deployment.yaml
  - kubectl rollout status deploy nginx

Жизненный цикл обновления приложения

- PROGRESSING
  - Deployment создает новый ReplicaSet
  - Deployment разворачивает новые ReplicaSet
  - Deployment уменьшает количество ReplicaSet(.
  - Новые Pod становятся ready или avaliable
- COMPLETE
  - Все реплики, ассоциированные с Deployment были обновлены до новой версии
  - Все реплики, ассоциированные с Deployment доступны
  - Нет старых запущенных реплик Deployment’а
- FAILED
  - Недостаточная квота
  - Readiness probe failure
  - Image pull error
  - Недостаточные права
  - Ошибка в конфигурации

Работа с хранилищами

1. Empty Dir
2. Host Path
3. Git Repo
4. Persistent volumes

Работа с хранилищами - EmptyDir

```
spec:
  containers:
  - image: gcr.io/google_containers/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

Работа с хранилищами - HOST PATH

```
spec:
  containers:
  - image: gcr.io/google_containers/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-pd
      name: test-volume
  volumes:
  - name: test-volume
    hostPath:
      # directory location on host
      path: /data
```

Работа с хранилищами – Git repo

```
spec:
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - mountPath: /mypath
      name: git-volume
  volumes:
  - name: git-volume
    gitRepo:
      repository: "git@somewhere:me/my-git-repository.git"
      revision: "22f1d8406d464b0c0874075539c1f2e96c253775"
```

Работа с хранилищами – Persistent volumes

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: slow
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://127.0.0.1:8081"
  clusterid: "630372ccdc720a92c681fb928f27b53f"
  restauthenabled: "true"
  restuser: "admin"
  secretNamespace: "default"
  secretName: "heketi-secret"
  gidMin: "40000"
  gidMax: "50000"
  volumetype: "replicate:3"
```

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator: In, values: [dev]}
```

Работа с хранилищами – Persistent volumes

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: slow
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://127.0.0.1:8081"
  clusterid: "630372ccdc720a92c681fb928f27b53f"
  restauthenabled: "true"
  restuser: "admin"
  secretNamespace: "default"
  secretName: "heketi-secret"
  gidMin: "40000"
  gidMax: "50000"
  volumetype: "replicate:3"
```

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator: In, values: [dev]}
```

ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  Name: example-configmap
data:
  # property-like keys
  game-properties-file-name: game.properties
  ui-properties-file-name: ui.properties
  # file-like keys
  game.properties: |
    enemies=aliens
    lives=3
    enemies.cheat=true
    enemies.cheat.level=noGoodRotten
    secret.code.passphrase=UUDDLRLRBABAS
    secret.code.allowed=true
    secret.code.lives=30
  ui.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
```

ConfigMap

- Из директории
  - kubectl create configmap nginx-config --from-file=/path/to/dir
  - ls /path/to/dir
  - kubectl describe configmaps nginx-config
- Из файла
  - kubectl create configmap nginx-config --from-file=<path-to-file>
  - kubectl describe configmaps nginx-config
  - kubectl create configmap nginx-config --from-file=<my-key-name>=<path-to-file>
- Из консоли
  - kubectl create configmap special-config --from-literal=special.how=very --from-literal=special.type=charm
  - kubectl describe configmaps nginx-config

Secrets

- SECRETS
  - Предназначены для хранения конфиденциальной информации
  - Может быть создан пользователем или системой
- CREATING SECRETS
  - Для создания используется команда ***kubectl create secret***
  - Создается вручную(в json или yaml формате и затем создает этот объект)
- USING SECRETS
  - Использование Secrets как файлов в Pod’ах
  - Использование Secrets как переменные окружения
  - Использование imagePullSecrets

Job

- Pod завершается успешно, *job* завершается успешно
- Удаление Job'ы удаляет Pod
- Job будет запускать новый Pod если предыдущий завершился с ошибкой
- A Job может использоваться для параллельного запуска multiple pod

DaemonSet

- Гарантирует, что все (или некоторые) узлы запускают копию pod
- Когда в кластер добавляется нода, на ней запускается pod
- Когда нода удаляется из кластера, pod удаляется с неё
- Удаление DaemonSet приводит к удалению всех pod’ов, которые запустил DaemonSet

Statefulsets

- StatefulSets предназначены для использования stateful приложениями и распределенными системами
- Предоставляет гарантированный порядок запуска

Statefulsets spec

```yaml
---
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: gcr.io/google_containers/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: my-storage-class
      resources:
        requests:
          storage: 1Gi
```

Thanks for Your Attention

Questions?

Introduction to Devops

Author: