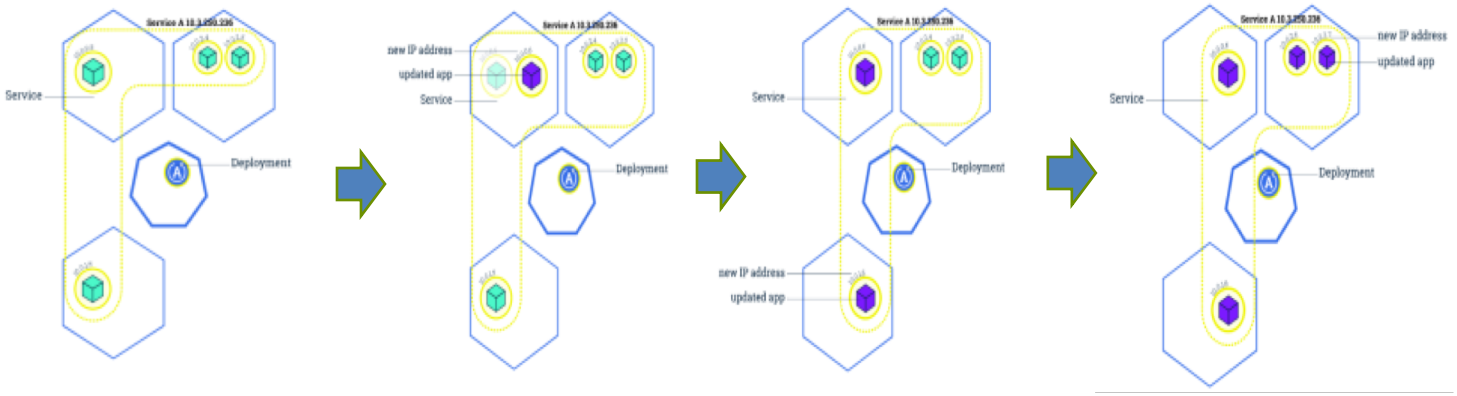
Chapter 5: Kubernetes for Continuous Integration

Kubernetes: Rolling Update

Updates the pods in a serial manner will load balancing traffic to available instances

- If deployment is exposed publicly, the service will load-balance traffic to only active and available pods
- Used to enable zero downtime updates
- Allows greater application update frequency
- Can also be leveraged to rollback to previous versions



Zero Downtime Rolling updates and rollback

$ kubectl set image <deployment> <new-image>

```
peter@Azure:~$ kubectl run nginx --image nginx:1.12.1 --port 80
deployment "nginx" created
peter@Azure:~$ kubectl set image deployments/nginx nginx=nginx:latest
deployment "nginx" image updated
peter@Azure:~$ kubectl describe deployment nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Fri, 15 Sep 2017 13:08:50 +0000
Labels:                 run=nginx
Annotations:            deployment.kubernetes.io/revision=2
Selector:               run=nginx
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  1 max unavailable, 1 max surge
Pod Template:
  Labels:       run=nginx
  Containers:
   nginx:
    Image:              nginx:latest
    Port:               80/TCP
    Environment:        <none>
    Mounts:             <none>
  Volumes:              <none>
Conditions:
  Type          Status  Reason
  ----          ------  ------
  Available     True    MinimumReplicasAvailable
OldReplicaSets: <none>
NewReplicaSet:  nginx-2688028062 (1/1 replicas created)
Events:
  FirstSeen     LastSeen        Count   From                    SubObjectPath   Type            Reason                  Message
  ---------     --------        -----   ----                    -------------   --------        ------                  -------
  32s           32s             1       deployment-controller                   Normal          ScalingReplicaSet       Scaled up replica set nginx-1295584306 to 1
  10s           10s             1       deployment-controller                   Normal          ScalingReplicaSet       Scaled up replica set nginx-2688028062 to 1
  10s           10s             1       deployment-controller                   Normal          ScalingReplicaSet       Scaled down replica set nginx-1295584306 to 0
```

Note: Rolling updates can also be undone (see next slide)

$ kubectl rollout undo <deployment>

```
peter@Azure:~$ kubectl rollout undo deployments/nginx
deployment "nginx" rolled back
peter@Azure:~$ kubectl describe deployments/nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Fri, 15 Sep 2017 13:36:49 +0000
Labels:                 run=nginx
Annotations:            deployment.kubernetes.io/revision=3
Selector:               run=nginx
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  1 max unavailable, 1 max surge
Pod Template:
  Labels:       run=nginx
  Containers:
   nginx:
    Image:              nginx:1.12.1
    Port:               80/TCP
    Environment:        <none>
    Mounts:             <none>
  Volumes:              <none>
Conditions:
  Type          Status  Reason
  ----          ------  ------
  Available     True    MinimumReplicasAvailable
OldReplicaSets: <none>
NewReplicaSet:  nginx-1295584306 (1/1 replicas created)
Events:
  FirstSeen     LastSeen        Count   From                    SubObjectPath   Type        Reason              Message
  ---------     --------        -----   ----                    -------------   --------    ------              -------
  1m            1m              1       deployment-controller                   Normal      ScalingReplicaSet   Scaled up replica set nginx-2688028062 to 1
  1m            1m              1       deployment-controller                   Normal      ScalingReplicaSet   Scaled down replica set nginx-1295584306 to 0
  1m            10s             2       deployment-controller                   Normal      ScalingReplicaSet   Scaled up replica set nginx-1295584306 to 1
  10s           10s             1       deployment-controller                   Normal      DeploymentRollback  Rolled back deployment "nginx" to revision 1
  10s           10s             1       deployment-controller                   Normal      ScalingReplicaSet   Scaled down replica set nginx-2688028062 to 0
```

Note: Rollbacks can also be also enacted with zero-downtime

```
root@ip-172-31-0-112:~/code# kubectl run nginx --image nginx:1.12.1 --port 80
deployment.apps/nginx created
root@ip-172-31-0-112:~/code# kubectl set image deployments/nginx nginx=nginx:latest
deployment.extensions/nginx image updated
root@ip-172-31-0-112:~/code# kubectl describe deployment nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Thu, 19 Jul 2018 04:00:55 +0000
Labels:                 run=nginx
Annotations:            deployment.kubernetes.io/revision=2
Selector:               run=nginx
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  run=nginx
  Containers:
   nginx:
    Image:        nginx:latest
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-6c486b77db (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  43s   deployment-controller  Scaled up replica set nginx-7f9bc86464 to 1
  Normal  ScalingReplicaSet  11s   deployment-controller  Scaled up replica set nginx-6c486b77db to 1
  Normal  ScalingReplicaSet  10s   deployment-controller  Scaled down replica set nginx-7f9bc86464 to 0
```

```
root@ip-172-31-0-112:~/code# kubectl rollout undo deployments/nginx
deployment.extensions/nginx
root@ip-172-31-0-112:~/code# kubectl describe deployments/nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Thu, 19 Jul 2018 04:00:55 +0000
Labels:                 run=nginx
Annotations:            deployment.kubernetes.io/revision=3
Selector:               run=nginx
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  run=nginx
  Containers:
   nginx:
    Image:        nginx:1.12.1
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-7f9bc86464 (1/1 replicas created)
Events:
  Type    Reason             Age              From                   Message
  ----    ------             ----             ----                   -------
  Normal  ScalingReplicaSet  1m               deployment-controller  Scaled up replica set nginx-6c486b77db to 1
  Normal  ScalingReplicaSet  1m               deployment-controller  Scaled down replica set nginx-7f9bc86464 to 0
  Normal  ScalingReplicaSet  18s (x2 over 2m) deployment-controller  Scaled up replica set nginx-7f9bc86464 to 1
  Normal  DeploymentRollback 18s              deployment-controller  Rolled back deployment "nginx" to revision 1
  Normal  ScalingReplicaSet  17s              deployment-controller  Scaled down replica set nginx-6c486b77db to 0
```

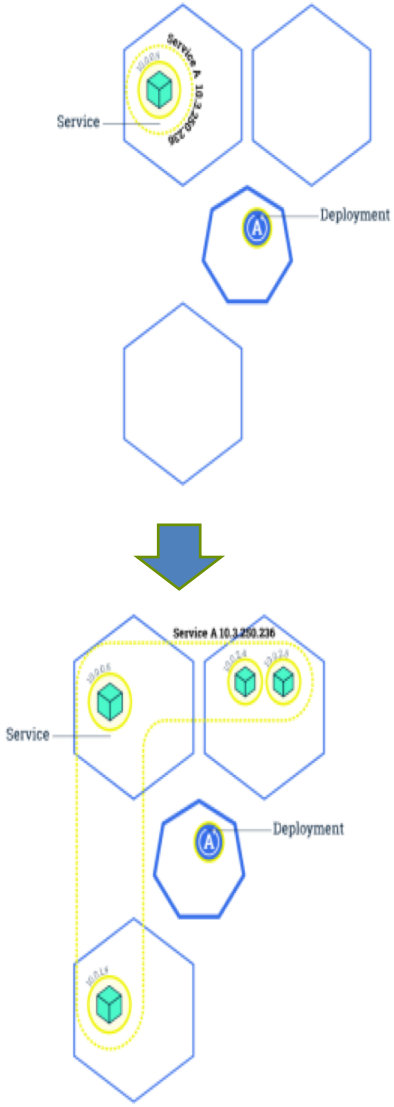
Note: Rollbacks can also be also enacted with zero-downtime
kubectl rollout status deployment nginx
kubectl rollout history deployment nginx
kubectl rollout history deployment/nginx --revision=3
kubectl rollout undo deployment/nginx --to-revision <num>

Zero Downtime Scaling

Kubernetes: Scaling

Changing the number of resources to meet a desired state

- Accomplished by adjusting the number of replicas in a deployment
- Accommodates both scaling up and scaling down to a minimum of 0
- Traffic is automatically sent to newly created instances through the service load balancer
- Can be used to enable rolling updates without downtime



$ kubectl scale deployments/<deployment> --replicas=<new num>

$ kubectl scale deploy/nginx --replicas=4

```
peter@Azure:~$ kubectl run nginx --image nginx --port 80
deployment "nginx" created
peter@Azure:~$
peter@Azure:~$ kubectl get deployments
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx     1         1         1            1           39s
peter@Azure:~$ kubectl scale deployments/nginx --replicas=2
deployment "nginx" scaled
peter@Azure:~$ kubectl get deployments
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx     2         2         2            1           1m
```

Note: Make sure to delete the deployments or replicasets when trying to clear out pods. Many a new user has repeated deleted pods and gotten frustrated when they respawn (as the deployments/replicasets are programed to do).

```
root@ip-172-31-0-112:~/code# kubectl run nginx --image nginx --port 80
deployment.apps/nginx created
root@ip-172-31-0-112:~/code#
root@ip-172-31-0-112:~/code# kubectl get deployments
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx              1         1         1            1           8s
nginx-deployment   3         3         3            3           17m
redis              1         1         1            1           33m
root@ip-172-31-0-112:~/code#
root@ip-172-31-0-112:~/code# kubectl scale deployments/nginx --replicas=3
deployment.extensions/nginx scaled
root@ip-172-31-0-112:~/code# kubectl get deployments
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx              3         3         3            3           34s
nginx-deployment   3         3         3            3           18m
redis              1         1         1            1           33m
root@ip-172-31-0-112:~/code# kubectl get pods
NAME                                READY     STATUS    RESTARTS   AGE
nginx-6f858d4d45-b4cv2              1/1       Running   0          42s
nginx-6f858d4d45-g42mg              1/1       Running   0          16s
nginx-6f858d4d45-l69jv              1/1       Running   0          16s
nginx-apparmor                      1/1       Running   0          30m
nginx-deployment-67594d6bf6-7v5lk   1/1       Running   0          12m
nginx-deployment-67594d6bf6-fdqx2   1/1       Running   0          18m
nginx-deployment-67594d6bf6-sx62p   1/1       Running   0          18m
redis-7869f8966-sq8xm               1/1       Running   0          33m
```

Autoscaling and HPA (Horizontal Pod Autoscaler)

Kubernetes: AutoScaling

HPA – Horizontal Pod Autoscaler:

Based on memory and CPU utilization by a pod, autoscaling scales up or down the number of pods

$ kubectl autoscale deployment nginx --min=2 --max=10

$ kubectl autoscale deployment nginx-deployment --max=5 --cpu-percent=80

$ kubectl get hpa

$ kubectl get hpa

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
nginx	Deployment/nginx	<unknown>/80%	1	4	3	12m
nginx-deployment	Deployment/nginx-deployment	<unknown>/80%	2	5	0	5s

Exercise 4 – (scaling, Auto scaling, rollout)

- For a previously created deployment scale the number of replicas
- see the difference in replicasets, pods , and see with a describe
- Rolling update, undo and see the changes in replica sets

cd to the ex4

https://github.com/shekhar2010us/kubernetes_teach_git/blob/master/ex4/scaling_rolling.md

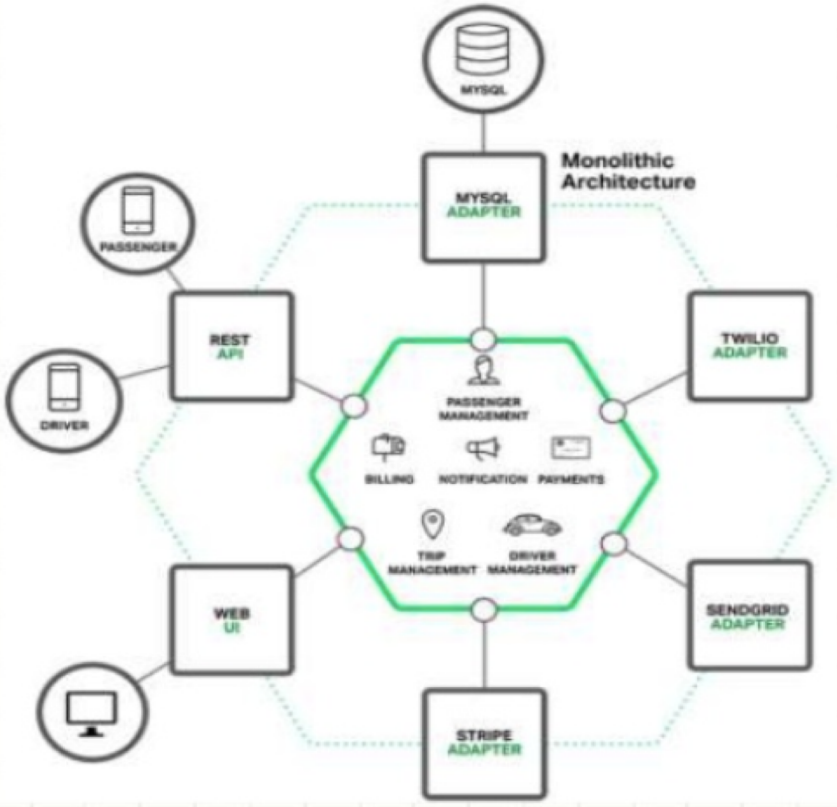
Service Discovery

Service Discovery

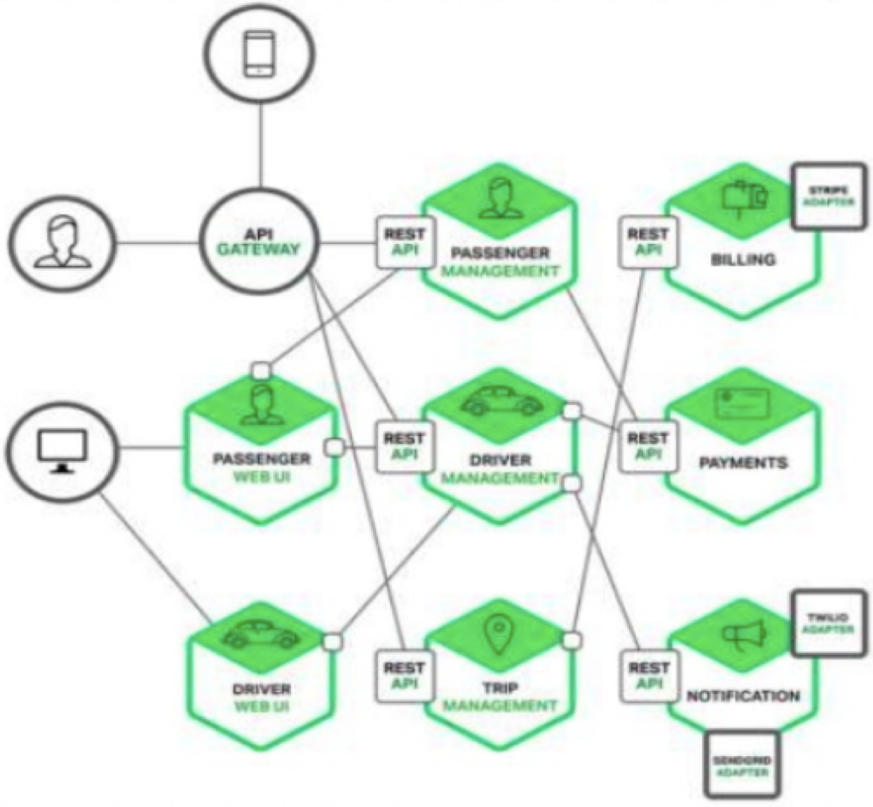
Migrating from a monolithic architecture to a Microservices based architecture where each service loosely coupled and horizontally scaled will require a service discovery mechanism to ensure that the REST API endpoints of the ephemeral services (say, docker containers running inside PODs) are resolved or detected. The IP addresses of the containers (or the PODs) being transient will need automated mechanism to resolve or convey the current IP addresses.

Sample Microservices Architecture

Monolith



Microservices



Service Discovery

So how do we keep track of all the microservices?

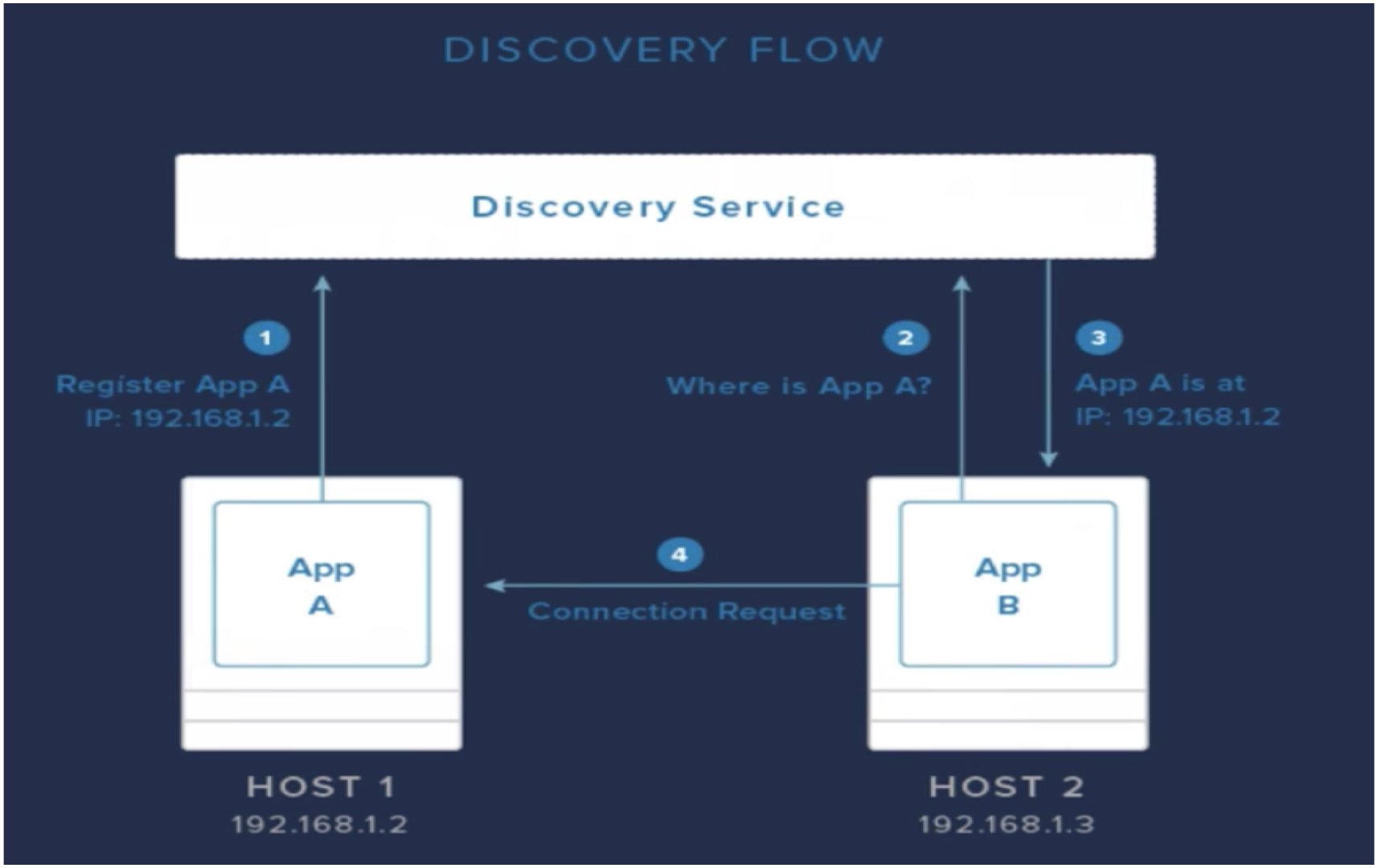
Service Discovery!

Service Discovery

What should Service Discovery provide?

- **Discover services** dynamically to get IP address and port details to communicate with other services
- **Health check**: Only healthy services should participate in handling traffic
- **Load balancing**: Traffic destined to a particular service should be dynamically load balanced

Service Discovery



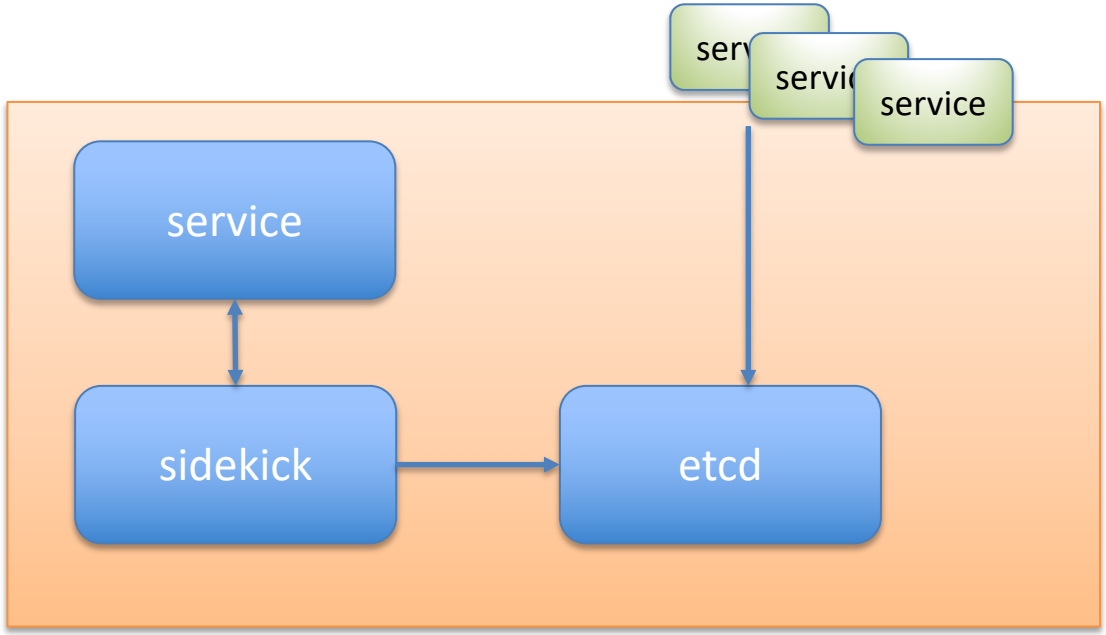
Service Discovery Tools

What are some Service Discovery tools?

- etcd, by CoreOS
- Consul, by Hashicorp
- Zookeeper, by Apache
- SkyDNS (built on top of etcd)
- Eureka, by Netflix
- Smartstack, by AirBnB

Service Discovery by etcd

- Etcd can be used as KV store for service registry
- Service itself can directly update etcd or a Sidekick service can be used to update etcd
- Sidekick service serves as **registrator**
- Other services can query etcd database to do the dynamic service discovery



Service Discovery by HashiCorp (Consul)

- Consul is a distributed, HA service discovery and configuration system
- Consul provides features - includes service and node discovery mechanisms, tagging system, health checks, consensus-based election routines, system-wide key/value storage, etc.
- Provides Rest based HTTP Api for external interaction

Service Discovery – Zookeeper by Apache

- Apache ZooKeeper is a distributed, open-source coordination service for distributed applications.
- Allows to read, write, and observe updates to data.
- Data are organized in a file system like hierarchy and replicated to all ZooKeeper servers in the ensemble (a set of ZooKeeper servers).
- Operations on data are atomic and sequentially consistent

Kubernetes Service Discovery

Using the environment variables that use the same conventions as those created by Docker links. Using DNS to resolve the service names to the service's IP address.

1. Environment Variables

Kubernetes injects environment variables for each service and each port exposed by the service. This makes it easy to deploy containers that use Docker links to find their dependencies.

For example, a RabbitMQ service can be located by using the RABBITMQ_SERVICE_SERVICE_HOST and RABBIT_MP_SERVICE_SERVICE_PORT variables.

Other environment variables are also exposed to support this.

Kubernetes Service Discovery

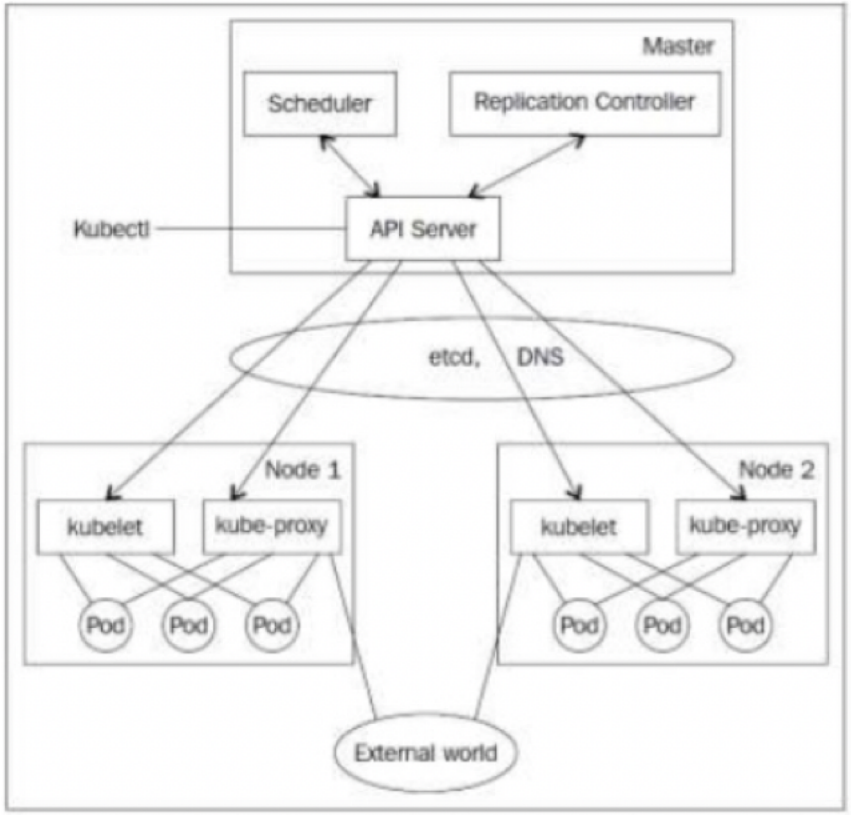
2. DNS Resolution

Kubernetes has a kube-dns addon that exposes the service's name as a DNS entry. As a result, you can tell your application to connect to a host name.

The service names are scoped within namespaces. This allows you to run different deployment of a service for each namespace (for example, one per developer or one per environments) without having to edit configuration files.

K8s Service Discovery Components

- **SkyDNS** – map Service name to IP address
- **Etcd** – KV store for service database
- **Kubelet** – healthcheck and replication controller takes care of maintaining pod count
- **Kube-proxy** – takes care of load balancing traffic to individual pods. Watches service changes and updates IPTables



Service Discovery

- **Services** – get mapped to pods using Selector labels. In the example: “MyApp” is the label
- **Service port** gets mapped to targetPort in the pod

```
{
"kind": "Service",
"apiVersion": "v1",
"metadata": {"name": "my-service"},
"spec": {
  "selector":{"app": "MyApp"},
  "ports":[
    {
      "protocol":"TCP",
      "port":80,
      "targetPort":9376
    }
  ]
 }
}
```

Lab: Service Discovery

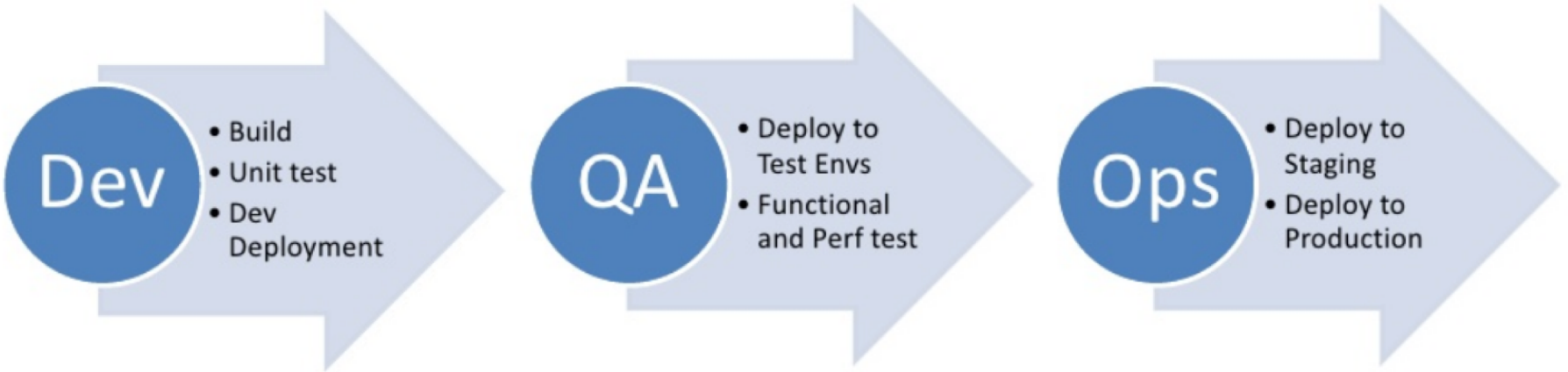
https://github.com/shekhar2010us/kubernetes_teach_git/blob/master/k8s_service_discovery/service-discovery.md

Example using consul:
https://github.com/hashicorp/demo-consul-101/tree/master/k8s

CICD (Jenkins) Integration with Kubernetes

Introduction to CI/CD

What is a Pipeline



Continuous Integration: Benefits

Reduced integration risk. More often than not, working on projects means multiple people are working on the separate tasks or parts of the code. The more people, the riskier the integration. Depending on how bad the problem really is, debugging and solving the issue can be really painful and can potentially mean a lot of changes to the code. Integrating on a daily basis or even more frequently can help reduce these kinds of problems to a minimum.

Higher code quality. Not having to worry about the problems, and focusing more on the functionality of the code results in a higher quality product.

The code in version control works. If you commit something that breaks the build, you and your team get the notice immediately and the problem is fixed before anyone else pulls the "broken" code.

Less time deploying. Deploying projects can be very tedious and time-consuming, and automating that process makes perfect sense

Continuous Integration: Requirements

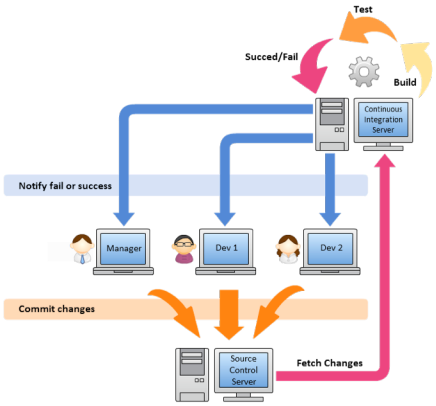
VCS: The first requirement is having the version control system (VCS). There is no way around it and there shouldn't be a way around it. VCS provides a reliable method to centralize and preserve changes made to your project over time. (Git)

CI Server: Continuous integration server (aka build server, aka CI server) is a software tool that centralizes all your CI operations and provides a reliable and stable environment for you to build your projects on.(Jenkins)

Jenkins

Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of software development process, with continuous integration and continuous delivery/continuos deployment.

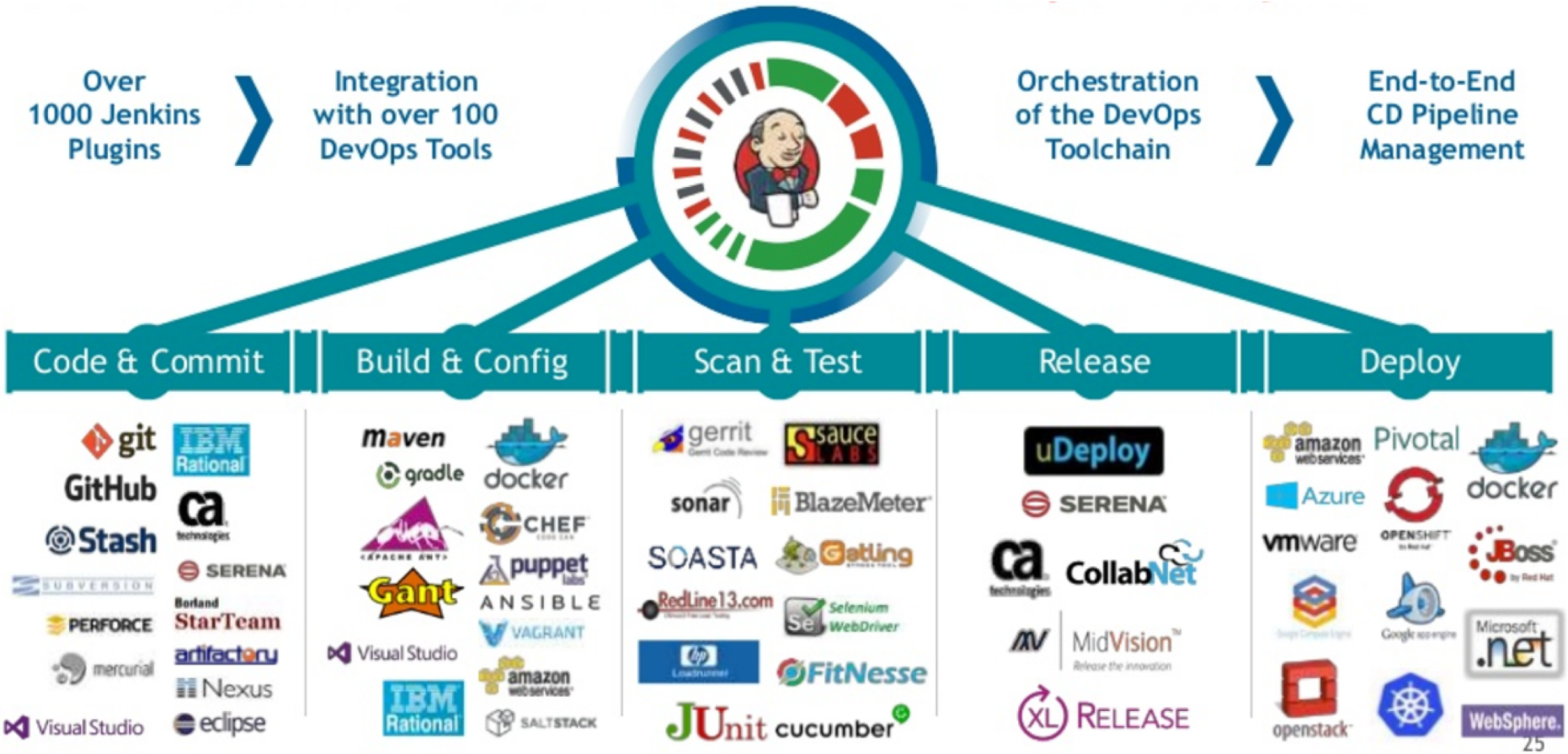
Jenkins



Continuous Integration

Continuous Delivery

Continuous Deployment



Deploy Jenkins in Kubernetes

https://github.com/jenkinsci/kubernetes-cd-plugin/blob/dev/README.md

https://medium.com/containerum/configuring-ci-cd-on-kubernetes-with-jenkins-89eab7234270

https://kubernetes.io/blog/2018/04/30/zero-downtime-deployment-kubernetes-jenkins/