

# CoreOS for SysAdmins

**Alex Crawford**

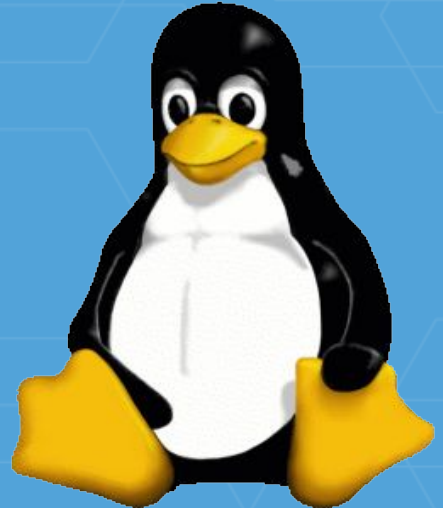
Software Developer at CoreOS

[alex.crawford@coreos.com](mailto:alex.crawford@coreos.com)

[github.com/crawford](https://github.com/crawford)

***What is CoreOS?***

# *What is CoreOS?*



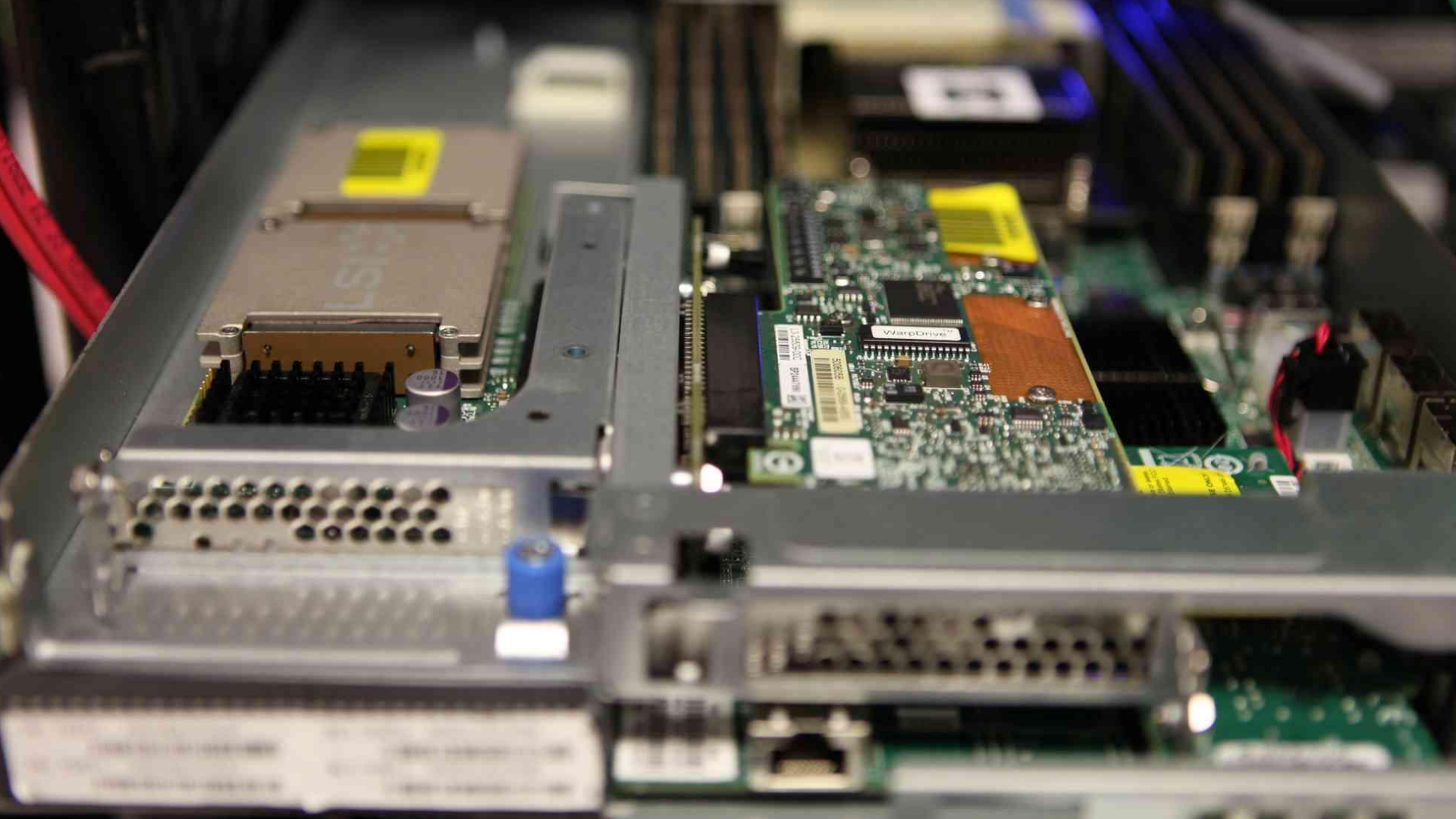


Google Cloud Platform



vmware®





# *What is CoreOS?*



# *What is CoreOS?*

 etcd

 flannel

 rkt





#### Top 100 Community Contributors



10k+ total commits  
~30 repositories  
324 releases  
22 CoreOS contributors  
605 community contributors

#### Top 100 Community Contributors







The smartest way to run your container infrastructure.

[tectonic.com](https://tectonic.com)    [@tectonic](https://twitter.com/tectonic)

# QUAY

Secure hosting for private container repositories

quay.io @quayio

# *Why build CoreOS?*

**The Datacenter as a Computer**  
*An Introduction to the Design of  
Warehouse-Scale Machines*

Luiz André Barroso and Urs Hölzle  
Google Inc.



**you**

**you as a sw engineer**

# your

```
with Ada.Text_IO;
```

```
procedure Hello_World is
```

```
  use Ada.Text_IO;
```

```
begin
```

```
  Put_Line("Hello, world!");
```

```
end;
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Hello, world!\n");
```

```
}
```

```
package main
```

```
import "fmt"
```

```
func main() {
```

```
    fmt.Println("Hello, world!")
```

```
}
```

**your**



**container  
image**



**your**



**/bin/java  
/opt/app.jar  
/lib/libc**

**your**



```
/bin/python  
/opt/app.py  
/lib/libc
```

**your**



**com.example.app**

**d474e8c57737625c**

**your**

Signed By: Alice

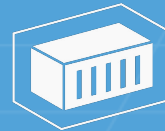
d474e8c57737625c



**ops engineer**

**you as an ops engineer**

**your**

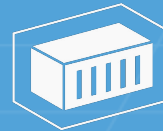




`com.example.webapp`  
x3



*your*



`com.example.webapp`  
x3



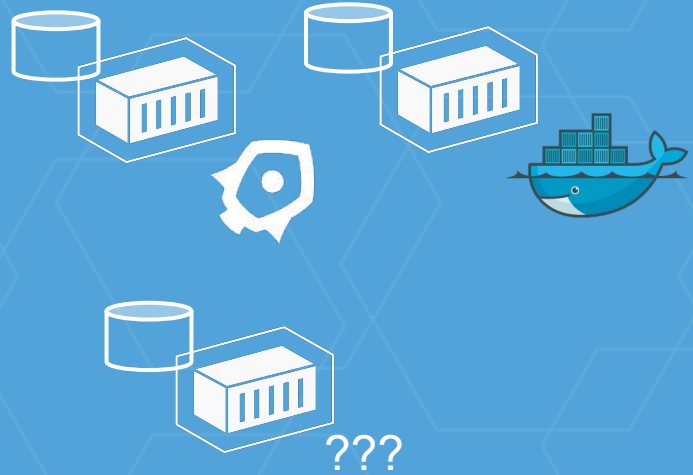
*your*



com.example.webapp  
x3



your



***How do we do it?***

*minimal*  
reduce API contracts

kernel  
systemd  
rkt  
ssh  
docker

python  
java  
nginx  
mysql  
openssl

stro distro distro distro distro

app

kernel  
systemd  
rkt  
ssh  
docker

stro distro distro distro distro

python  
java  
nginx  
mysql  
openssl

app



kernel  
systemd  
rkt  
ssh  
docker

stro distro distro distro distro

python  
openssl-A

app1

java  
openssl-B

app2

java  
openssl-B

app3

# CoreOS

distro distro distro distro distro

python  
openssl-A

app1

java  
openssl-B

app2

java  
openssl-B

app3

# CoreOS

stro distro distro distro distro

container

container

container

 rkt



# *OS operations*

The background is a solid blue color with a subtle, repeating pattern of light blue hexagons of varying sizes, creating a textured effect.

# *OS operations* updates

# manual updates





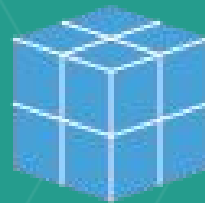
# automatic updates



# automatic updates



Data



Update

A

B



**Data**

**A**

**B**



# *CoreOS Updates*

atomic update with rollback

# *OS operations* machine configuration

***machine config***  
get into the cluster

[Service]

```
ExecStart=/usr/bin/kubelet --  
api_servers=https://172.17.4.101 --  
register-node=true --hostname-  
override=172.17.4.201 --cluster_dns=10.  
3.0.10 --cluster_domain=cluster.local  
--tls-cert-file=worker.pem --tls-  
private-key-file=worker-key.pem
```



[Service]

ExecStart=/usr/bin/kubelet --

**api\_servers=https://172.17.4.101** --

register-node=true --hostname-

override=172.17.4.201 --cluster\_dns=10.

3.0.10 --cluster\_domain=cluster.local

--tls-cert-file=worker.pem --tls-

private-key-file=worker-key.pem

[Service]

```
ExecStart=/usr/bin/kubelet --  
api_servers=https://172.17.4.101 --  
register-node=true --hostname-  
override=172.17.4.201 --cluster_dns=10.  
3.0.10 --cluster_domain=cluster.local  
--tls-cert-file=worker.pem --tls-  
private-key-file=worker-key.pem
```

The background is a solid blue color with a subtle, repeating pattern of light blue hexagons of varying sizes and orientations.

# *cluster operations*

## distributed configuration



# etcd

/etc

**distributed**

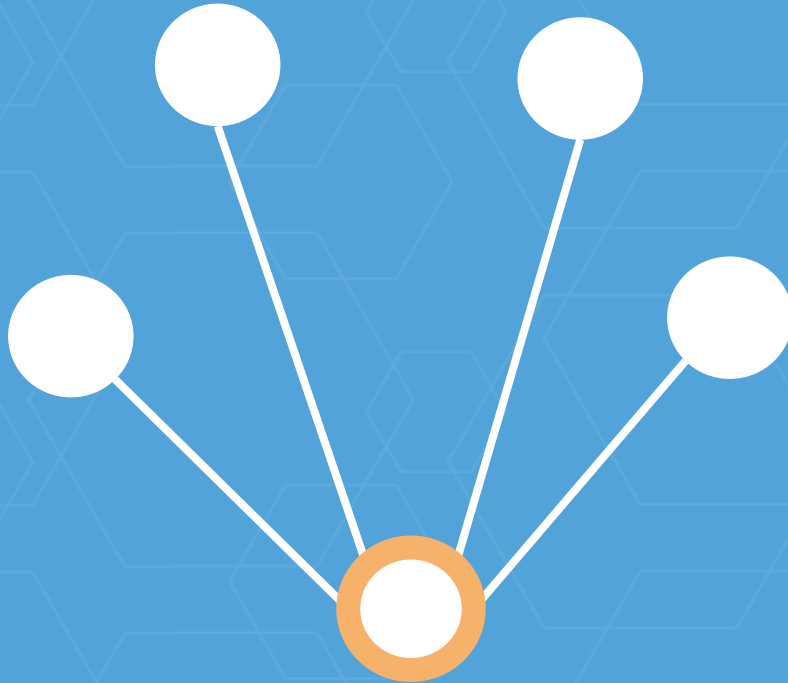
# Available



Leader



Follower



# Available



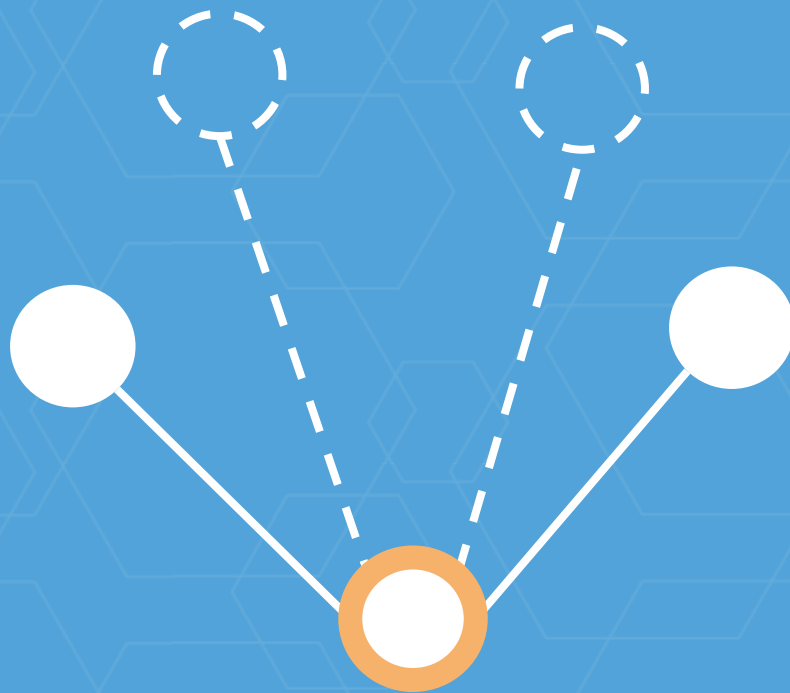
Leader



Follower



# Available





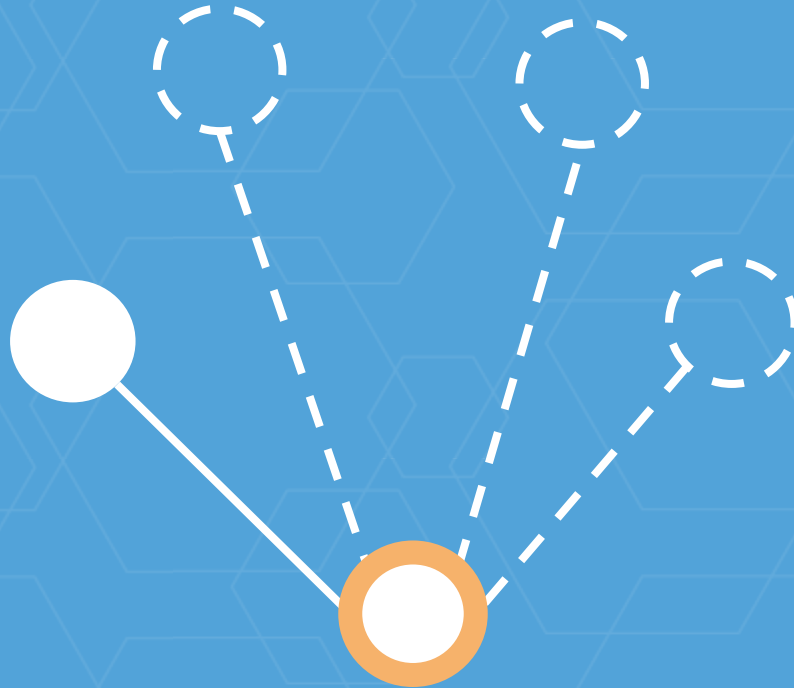
# Unavailable



Leader



Follower



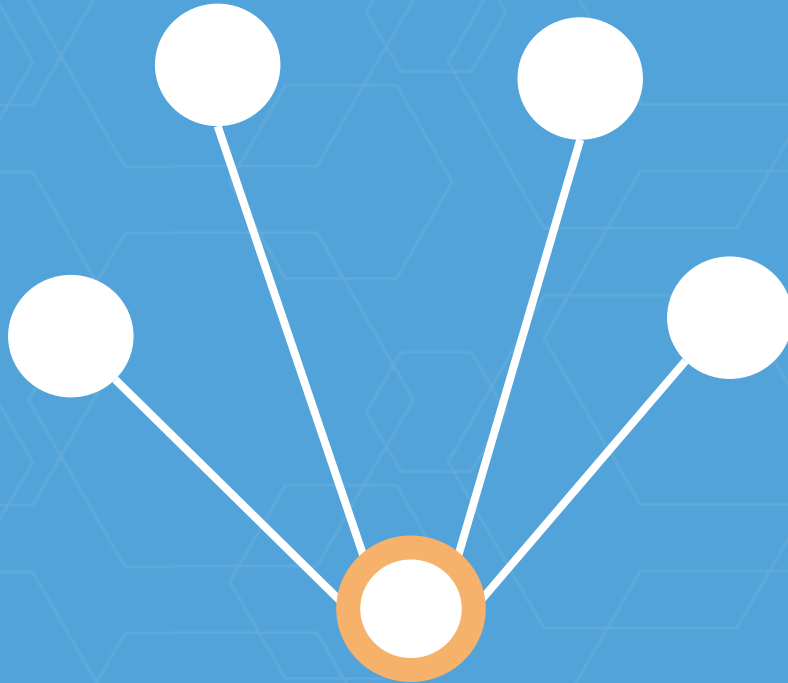
# Available



Leader



Follower



# Available



Leader



Follower



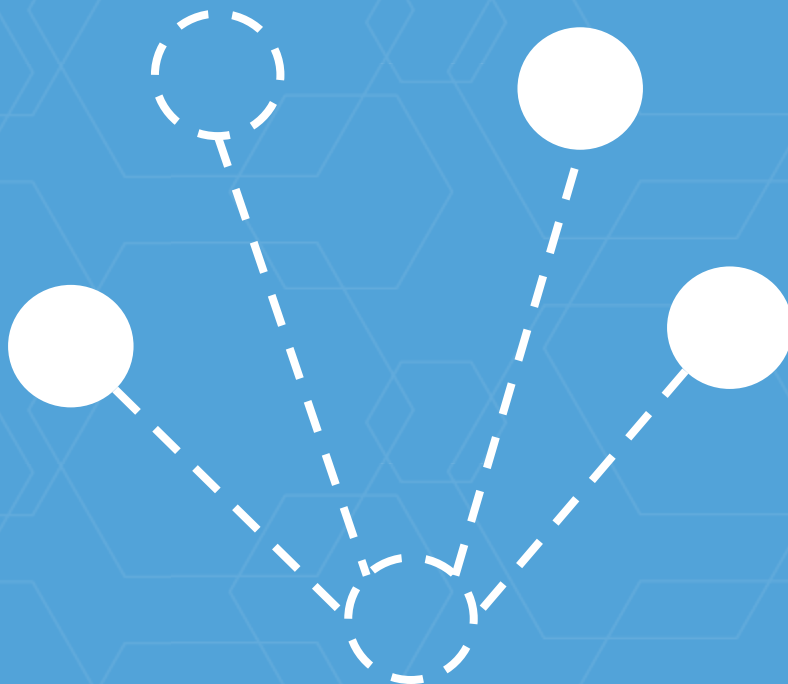
# Temporarily Unavailable



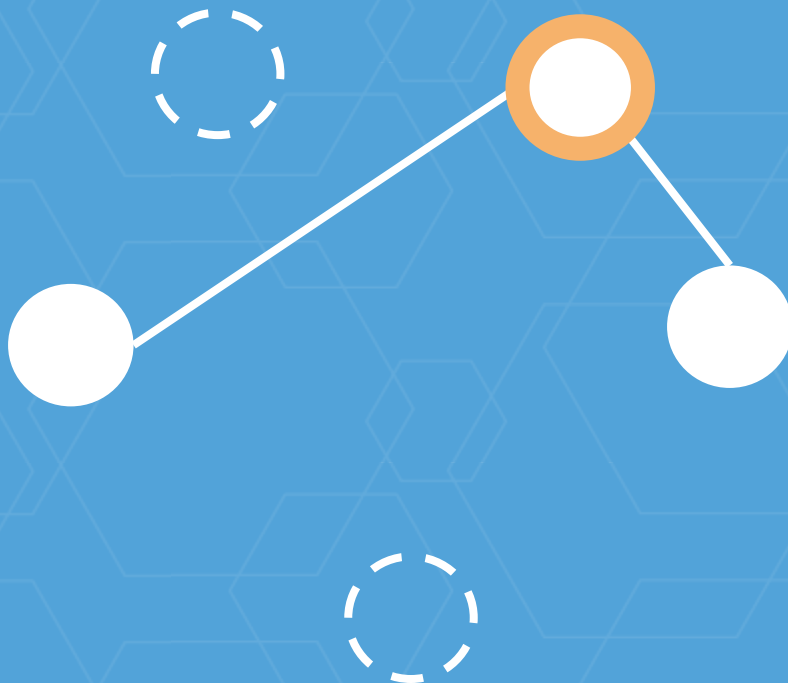
Leader



Follower



# Available



# Unavailable





# *cluster operations*

## what should run

*scheduling*

k8s/mesos/etc scheduler



*scheduling*  
getting work to servers

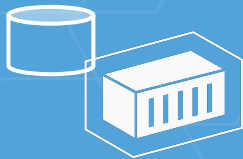
```
$ scp app host:/opt
```

```
$ ssh host systemd-run /opt/app
```



```
$ scp app host:/opt
```

```
$ ssh host systemd-run /opt/app
```



# \$ fab deploy:app



# \$ fab deploy:app



# \$ fab deploy:app



# \$ fab deploy:collector-app



# \$ fab deploy:collector-app





# \$ fab deploy:collector-app



# \$ fab lowest-loadaverage



\$ fab lowest-loadaverage  
host1



```
$ fab lowest-loadaverage
```

```
host1
```

```
$ fab -H host1 deploy:job
```



The background is a solid blue color with a repeating pattern of white-outlined hexagons of various sizes, creating a honeycomb-like texture.

# You

You



Scheduler API

You



Scheduler API



Scheduler

You



Scheduler API



Scheduler



Machine(s)



```
while true {  
    todo = diff(desState, curState)  
    schedule(todo)  
}
```

```
while true {  
  todo = diff(desState, curState)  
  schedule(todo)  
}
```

```
while true {  
  todo = diff(desState, curState)  
  schedule(todo)  
}
```

```
while true {  
  todo = diff(desState, curState)  
  schedule(todo)  
}
```

```
$ kubectl run example  
--image=quay.io/crawford/example  
--replicas=1
```

```
$ kubectl get pods
```

POD	IP
example-97wt8	10.2.29.4

```
$ kubectl run example  
--image=quay.io/crawford/example  
--replicas=1
```

```
$ kubectl get pods  
POD          IP  
example-97wt8 10.2.29.4
```

```
$ kubectl run example  
--image=quay.io/crawford/example  
--replicas=1
```

```
$ kubectl get pods
```

POD	IP
example-97wt8	10.2.29.4

```
$ kubectl run example  
--image=quay.io/crawford/example  
--replicas=1
```

```
$ kubectl get pods
```

POD	IP
example-97wt8	10.2.29.4



```
$ kubectl run example  
--image=quay.io/crawford/example  
--replicas=1
```

```
$ kubectl get pods
```

POD	IP
example-97wt8	10.2.29.4

```
$ kubectl run example  
--image=quay.io/crawford/example  
--replicas=1
```

```
$ kubectl get pods
```

POD

IP

<b>example-97wt8</b>	<b>10.2.29.4</b>
----------------------	------------------

```
$ kubectl scale rc example  
--replicas=2
```

```
$ kubectl get pods
```

POD	IP
example-97wt8	10.2.29.4
example-f839d	10.2.29.8

```
$ kubectl scale rc example  
--replicas=2
```

```
$ kubectl get pods
```

POD	IP
example-97wt8	10.2.29.4
example-f839d	10.2.29.8

```
$ kubectl scale rc example  
--replicas=2
```

```
$ kubectl get pods
```

POD	IP
example-97wt8	10.2.29.4
example-f839d	10.2.29.8

```
$ kubectl scale rc example  
--replicas=2
```

```
$ kubectl get pods
```

POD	IP
example-97wt8	10.2.29.4
example-f839d	10.2.29.8

```
$ kubectl scale rc example  
--replicas=2
```

```
$ kubectl get pods
```

POD	IP
<b>example-97wt8</b>	<b>10.2.29.4</b>
example-f839d	10.2.29.8

```
$ kubectl scale rc example  
--replicas=2
```

```
$ kubectl get pods
```

POD	IP
example-97wt8	10.2.29.4
<b>example-f839d</b>	<b>10.2.29.8</b>



**rc web-prod**

```
select(env=prod,app=web)  
count=3
```



**pod**

```
env=prod  
app=web
```

```
rc web-prod  
select(env=prod,app=web)  
count=1
```



```
pod  
env=prod  
app=web
```

**rc web-prod**

```
select(env=prod,app=web)  
count=1
```



**pod**

```
env=prod  
app=web
```

**rc web-prod**

select(env=prod,app=web)

**count=5**



**pod**

env=prod

app=web

**rc web-prod**

```
select(env=prod,app=web)  
count=5
```



**pod**

```
env=prod  
app=web
```

*cluster operations*  
where is it running

***services***  
dns, LBs, k8s labels



# *k8s labels* flexible service discovery



**pod**  
env=dev  
app=web

**pod**  
env=test  
app=web

**pod**  
env=prod  
app=web

**service test.example.com**  
select(env=dev,app=web)

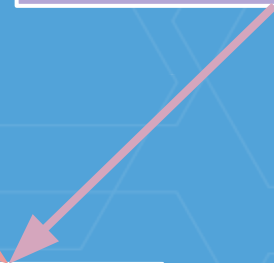
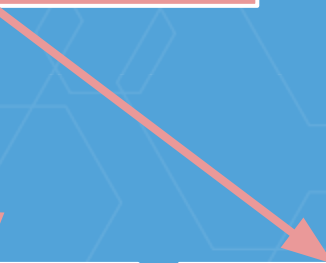
**service beta.example.com**  
select(env=test,app=web)  
OR  
select(env=prod,app=web)

**service example.com**  
select(env=prod,app=web)

**pod**  
env=dev  
app=web

**pod**  
env=test  
app=web

**pod**  
env=prod  
app=web



**service test.example.com**  
select(env=dev,app=web)

**service beta.example.com**  
select(env=test,app=web)  
OR  
select(env=prod,app=web)

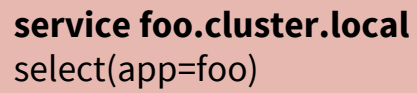
**service example.com**  
select(env=prod,app=web)

**pod**  
env=dev  
app=web

**pod**  
env=test  
app=web

**pod**  
env=prod  
app=web

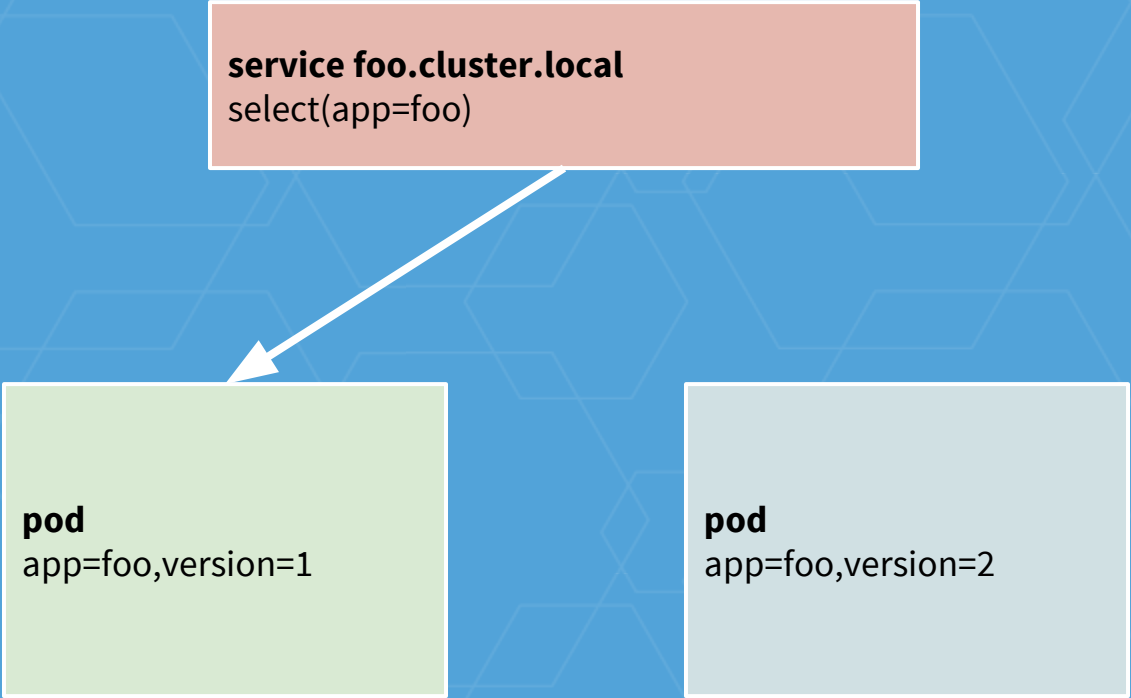
**service foo.cluster.local**  
select(app=foo)



A diagram illustrating a Kubernetes service selection. A red box at the top contains the text 'service foo.cluster.local' and 'select(app=foo)'. A white arrow points from this box to a green box at the bottom. The green box contains the text 'pod' and 'app=foo,version=1'. The background is blue with a faint hexagonal pattern.

**pod**  
app=foo,version=1

**service foo.cluster.local**  
select(app=foo)



**pod**  
app=foo,version=1

**pod**  
app=foo,version=2

**service foo.cluster.local**  
select(app=foo)

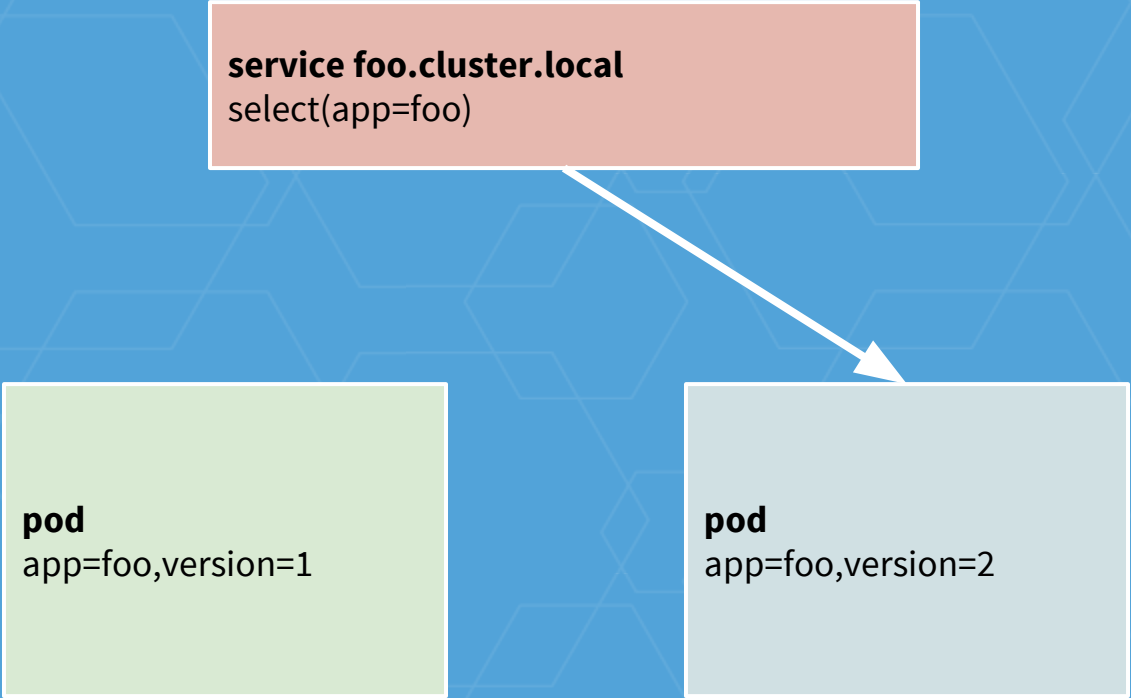
```
graph TD; S["service foo.cluster.local  
select(app=foo)"] --> P1["pod  
app=foo,version=1"]; S --> P2["pod  
app=foo,version=2"]
```

The diagram illustrates a Kubernetes service selection process. At the top, a red box represents the service 'foo.cluster.local' with the selector 'select(app=foo)'. Two white arrows point from this service box to two pod boxes below. The left pod box is green and represents 'pod app=foo,version=1'. The right pod box is light blue and represents 'pod app=foo,version=2'. This shows that the service successfully selects both pods because they both have the 'app=foo' label.

**pod**  
app=foo,version=1

**pod**  
app=foo,version=2

**service foo.cluster.local**  
select(app=foo)

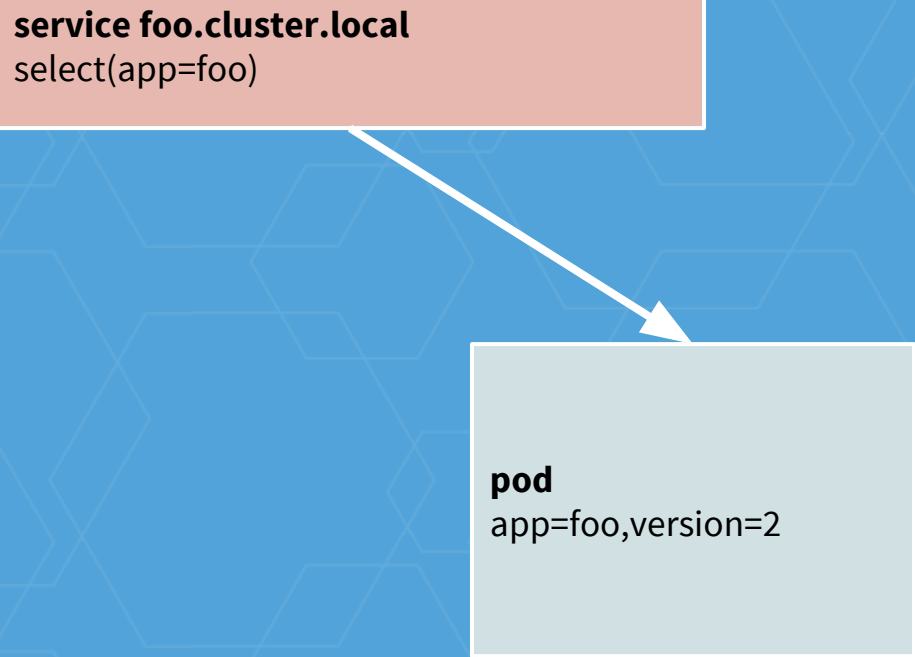


```
graph TD; S["service foo.cluster.local<br/>select(app=foo)"] --> B["pod<br/>app=foo,version=2"]; G["pod<br/>app=foo,version=1"]
```

**pod**  
app=foo,version=1

**pod**  
app=foo,version=2

**service foo.cluster.local**  
select(app=foo)



```
graph TD; S["service foo.cluster.local  
select(app=foo)"] --> P["pod  
app=foo,version=2"]
```

The diagram illustrates a Kubernetes service selection process. A red box at the top represents a service named 'foo.cluster.local' with a selector 'select(app=foo)'. A white arrow points from this service box to a light blue box below it, representing a pod with labels 'app=foo,version=2'. This visualizes how the service finds and routes traffic to the matching pod.

**pod**  
app=foo,version=2





The background is a solid blue color with a pattern of light blue, semi-transparent hexagons of varying sizes and orientations, creating a geometric, honeycomb-like texture.

***work with us***

[coreos.com/careers](https://coreos.com/careers)



**Questions?**