

```
$ kubectl
```

```
  --image
```

```
  --replicas=3
```



**WHAT HAPPENS
WHEN K8S JOURNEY**



@nnao45

Naoya Yokoyama



CyberAgent Inc.
Infra/ServerSide Engineer



Tech Advisor Startup Company



Zsh,BGP,Go,Rust,MySQL,K8S,AWS,Ansible



Vtuber,Game,Tennis



@nnao45,n4sekai5y@gmail.com



CNDJP

MySQLの商用版使いたい人生だった

DynamoDBのインデックス設計つらたん

RustのGraphDBのライブラリかきたい



V言語って最強の静的型付言語なの？

ぶいちゅーばー友達募集！

AGENDA

```
$ kubectl version -o json | jq '.clientVersion.gitVersion'  
"v1.13.4"  
$ kubectl version -o json | jq '.serverVersion.gitVersion'  
"v1.13.4"
```



AGENDA

- AUTH JOURNEY
- CONTROLLER LOOP
- POD DEPLOY



AUTH JOURNY

🤔 What happens when I type `kubectl run`?

```
$ kubectl run --image=nginx --replicas=3
```



AUTH JOURNY

The screenshot shows a GitHub repository page for 'jamiehannaford / what-happens-when-k8s'. At the top, there are navigation links for 'Code', 'Issues 6', 'Pull requests 5', 'Projects 0', 'Wiki', and 'Insights'. On the right, there are buttons for 'Watch 64', 'Unstar 1,212', and 'Fork 114'. Below the repository name, there is a question mark icon and the text 'What happens when I type kubectl run?'. A summary bar shows '27 commits', '1 branch', '0 releases', and '6 contributors'. Below this, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find File', and 'Clone or download'. A commit history section shows a merge pull request #8 from jinsenglin/patch-1 by jamiehannaford, with the latest commit e1820fb on 23 Nov 2018. Below the commit history, there is a file list with 'README.md' and a description 'Merge pull request #8 from jinsenglin/patch-1' dated '4 months ago'. The main content area shows the 'README.md' file with the following text:

What happens when ... Kubernetes edition!

Imagine I want to deploy nginx to a Kubernetes cluster. I'd probably type something like this in my terminal:

```
kubectl run --image=nginx --replicas=3
```

and hit enter. After a few seconds, I should see three nginx pods spread across all my worker nodes. It works like magic, and that's great! But what's really going on under the hood?

One of the awesome things about Kubernetes is that it handles the deployment of workloads across infrastructure through user-friendly APIs. Complexity is hidden by simple abstractions. But in order to fully understand the value it offers us, it's also useful to understand its internals. This guide will lead you through the full lifecycle of a request from the client to the kubelet, linking off to the source code where necessary to illustrate what's going on.

This is a living document. If you spot areas that can be improved or rewritten, contributions are welcome!



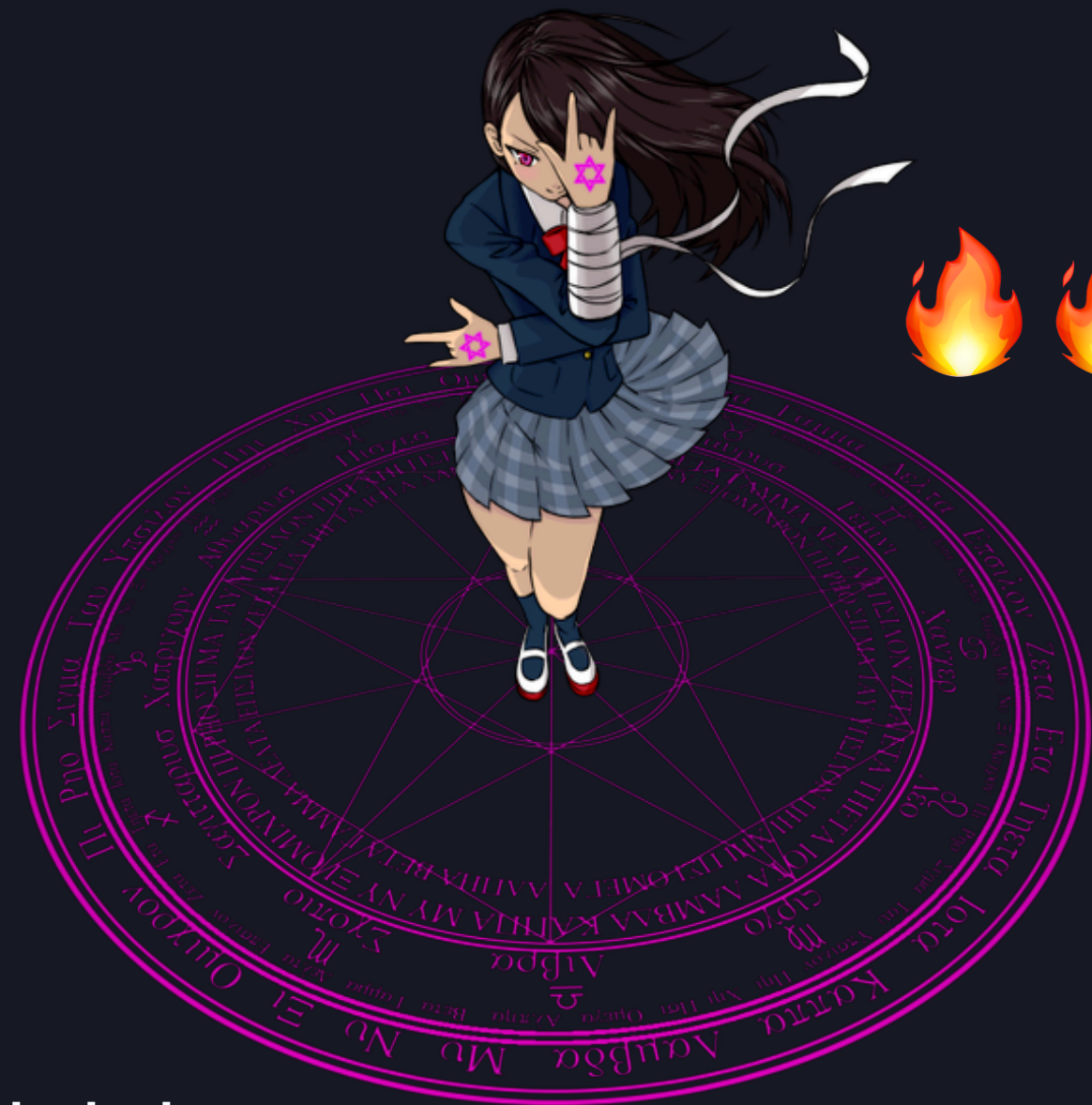
NEXT...



AUTH JOURNEY



AUTH JOURNEY



たたかう

👉 `kubectl run --image=nginx --replicas=3`

`kubectl get pod --all-namespaces`

`sudo rm -rf / --preserve-root`

コンテナをやめる

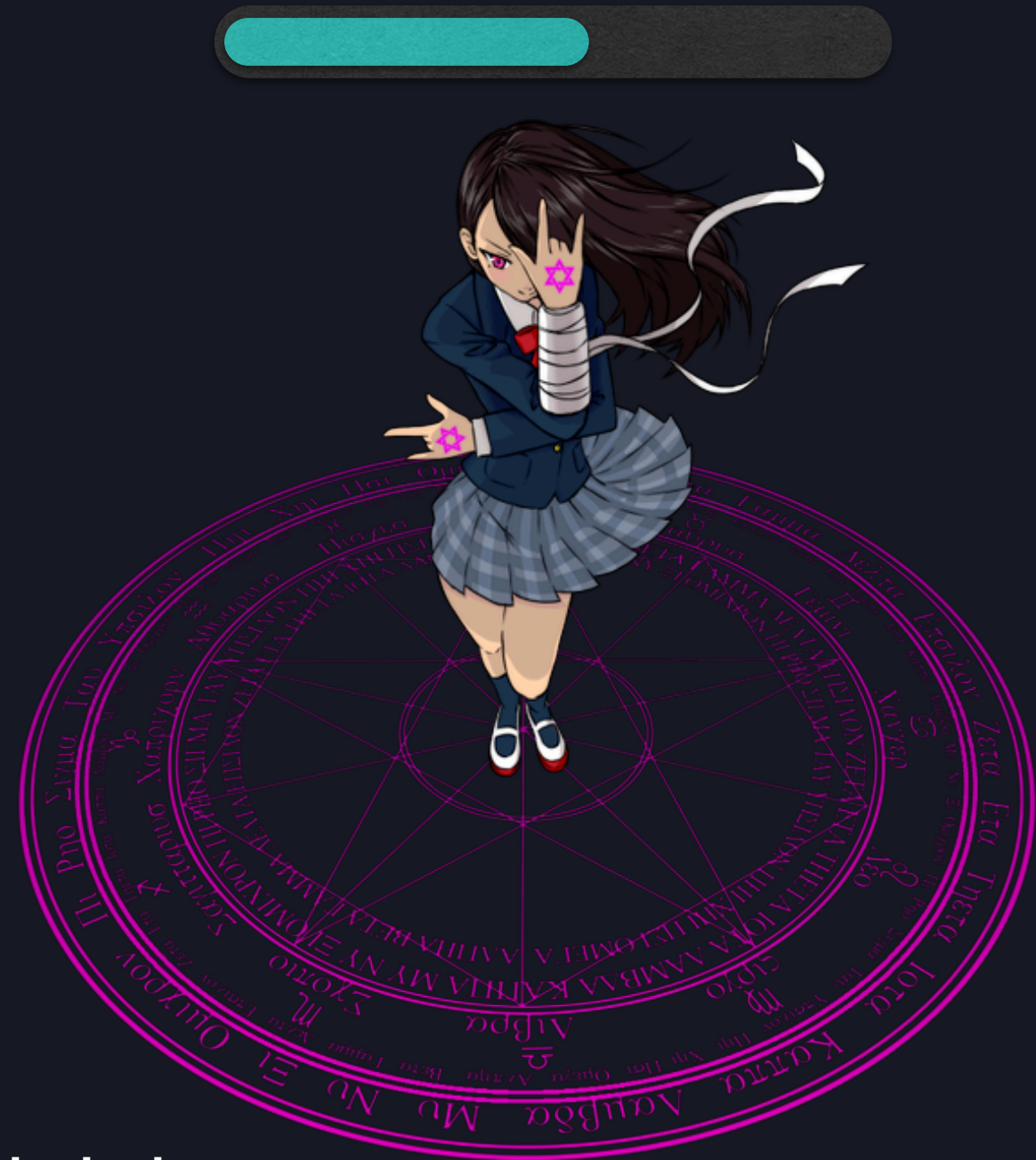


AUTH JOURNY

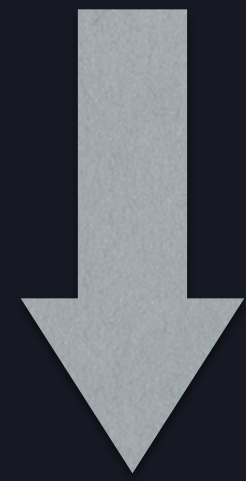
BEFORE FIRE... 



AUTH JOURNEY



詠唱中



BEFORE FIRE...

たたかう

👉 `kubectl run --image=nginx --replicas=3`

`kubectl get pod --all-namespaces`

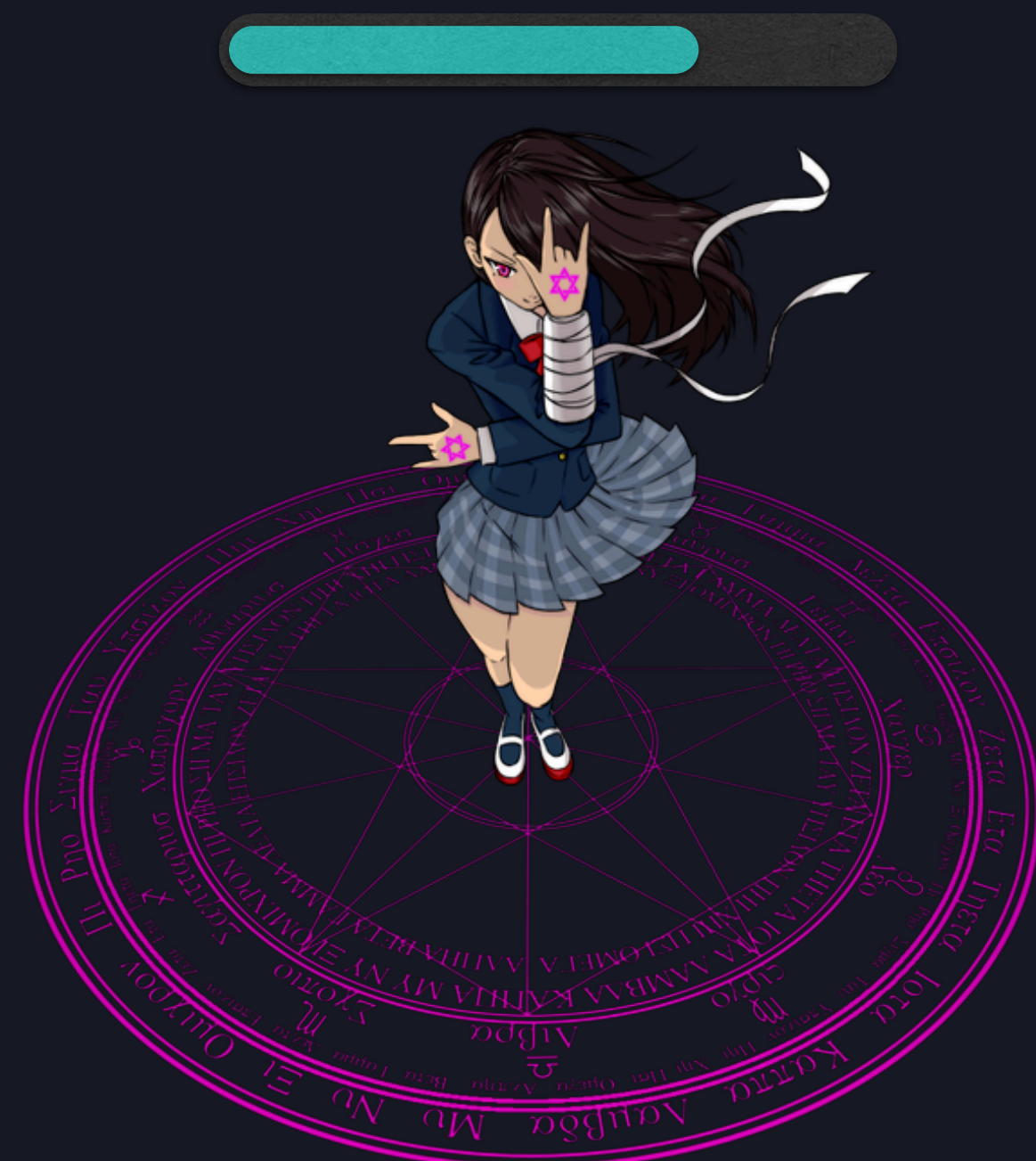
`sudo rm -rf / --preserve-root`

コンテナをやめる



AUTH JOURNEY

BEFORE FIRE...



1. RESOURCE VALIDATE

SEE `$ kubectl api-resources`

2. LOAD API SCHEMA

LOOK `~/.kube/cache/discovery/`

3. GENERATE REQUEST

たたかう

👉 `kubectl run --image=nginx --replicas=3`

`kubectl get pod --all-namespaces`

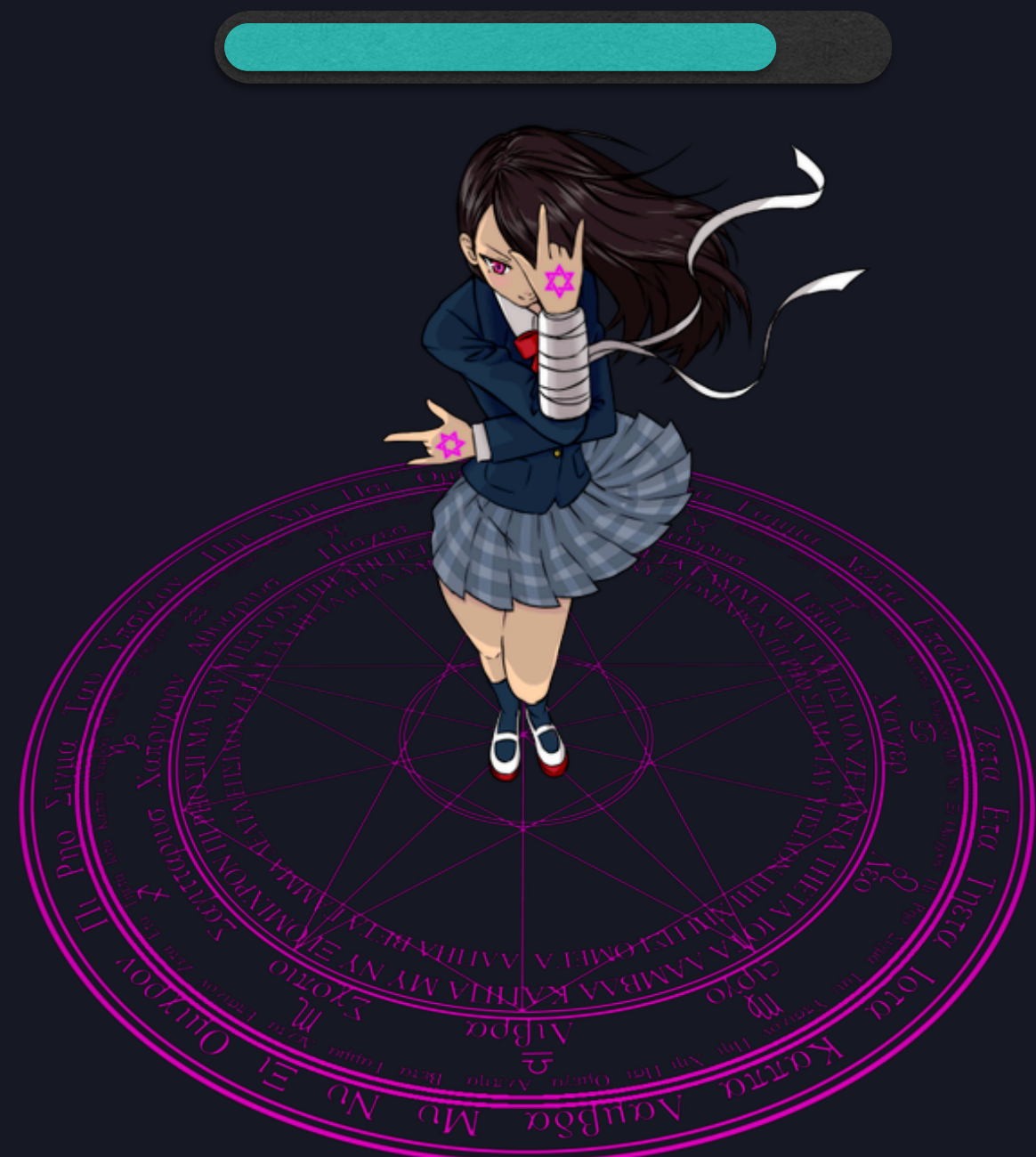
`sudo rm -rf / --preserve-root`

コンテナをやめる



AUTH JOURNEY

BEFORE FIRE...



たたかう

👉 `kubectl run --image=nginx --replicas=3`

`kubectl get pod --all-namespaces`

`sudo rm -rf / --preserve-root`

コンテナをやめる

4. LOOK KUBECONFIG CHECK PRIORITY

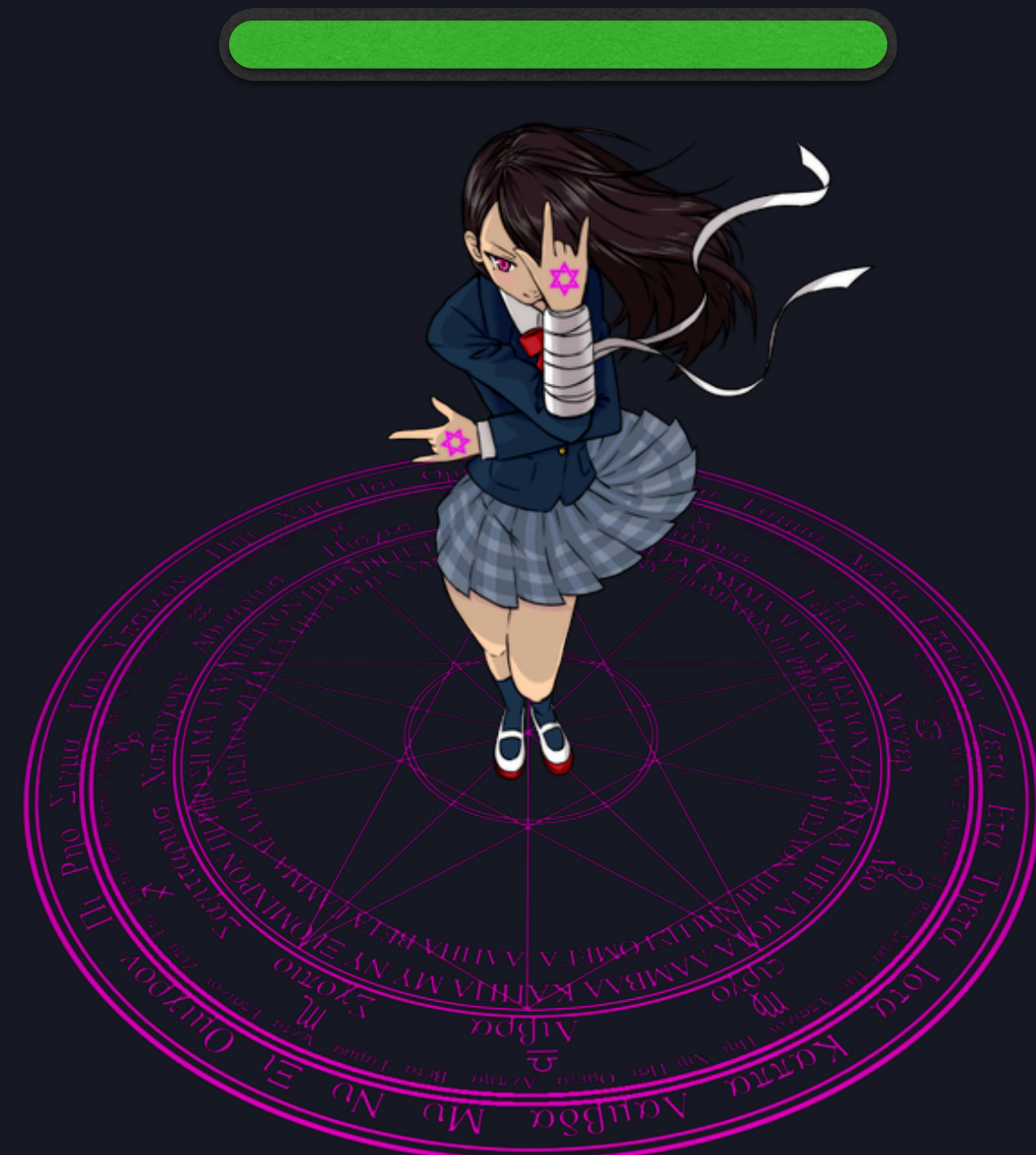
1.USE `$ kubectl --kubeconfig`

2.USE `$ ${KUBECONFIG} kubectl`

3.LOOK `~/kube or something`



AUTH JOURNEY



FIRE!!!!!!!!!!!!!!!



たたかう

👉 `kubectl run --image=nginx --replicas=3`

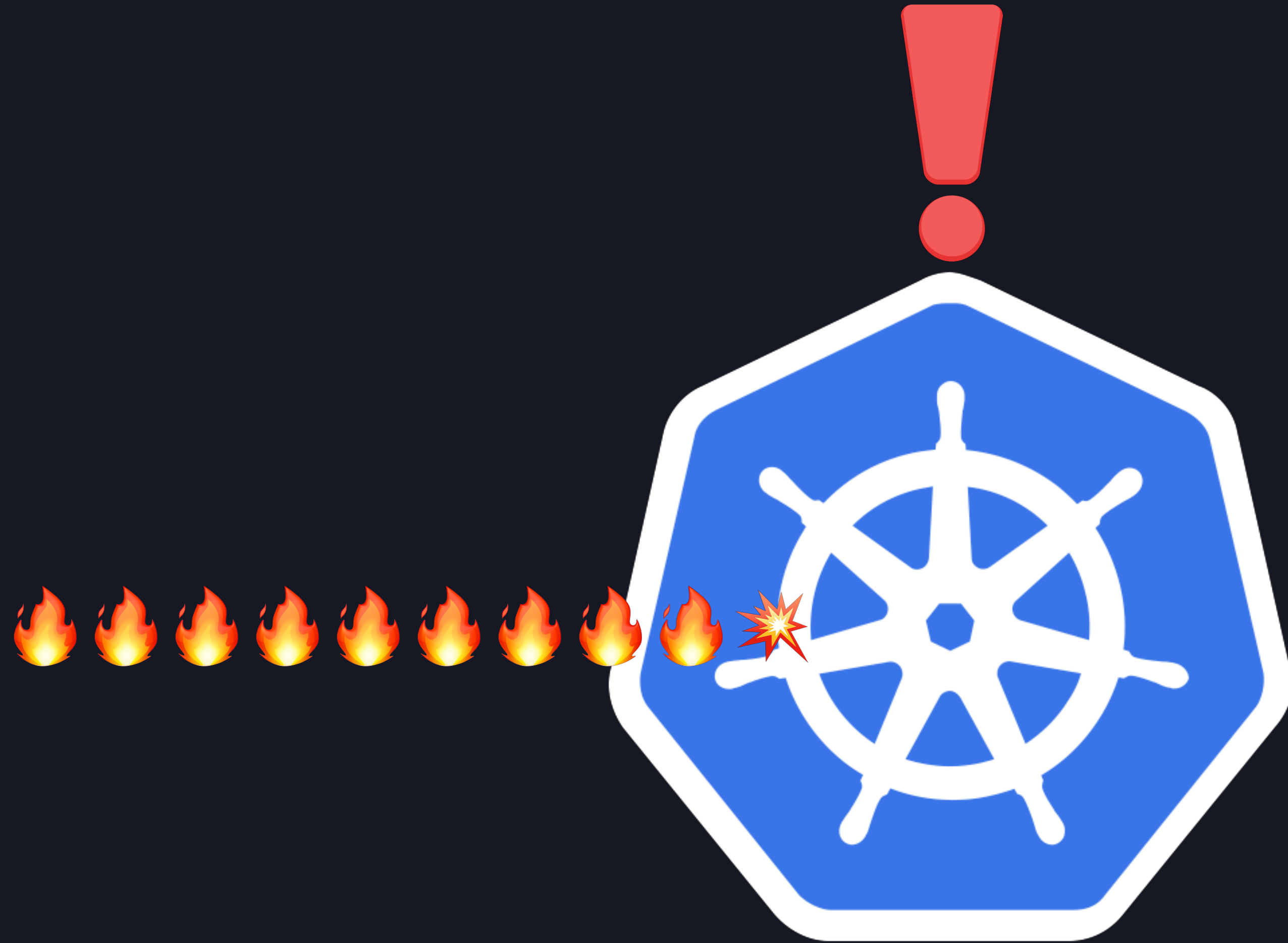
`kubectl get pod --all-namespaces`

`sudo rm -rf / --preserve-root`

コンテナをやめる



AUTH JOURNEY



AUTH JOURNEY

Authentication

接続元が確かに
用意されたアカ
ウントのクライ
アントかどうか



Authorization

Admission Control



AUTH JOURNEY

Authentication

kube-apiserver



Use...

X509

or

bearer

or

basic



AUTH JOURNEY

Authentication Method

X509

Validate Client TLS Key from CA ROOT Certificate

```
$ curl -key client.key -cert client.crt -cacert ca.crt
```

bearer

Validate Authorization Header

```
$ curl -H 'Authorization:Bearer xxxxx...' --cacert ...
```

basic

Validate Basic Auth

```
$ curl -u 'admin-user:admin-passwd'
```



AUTH JOURNEY

Authentication

接続元が確かに
用意されたアカ
ウントのクライ
アントかどうか

Authorization

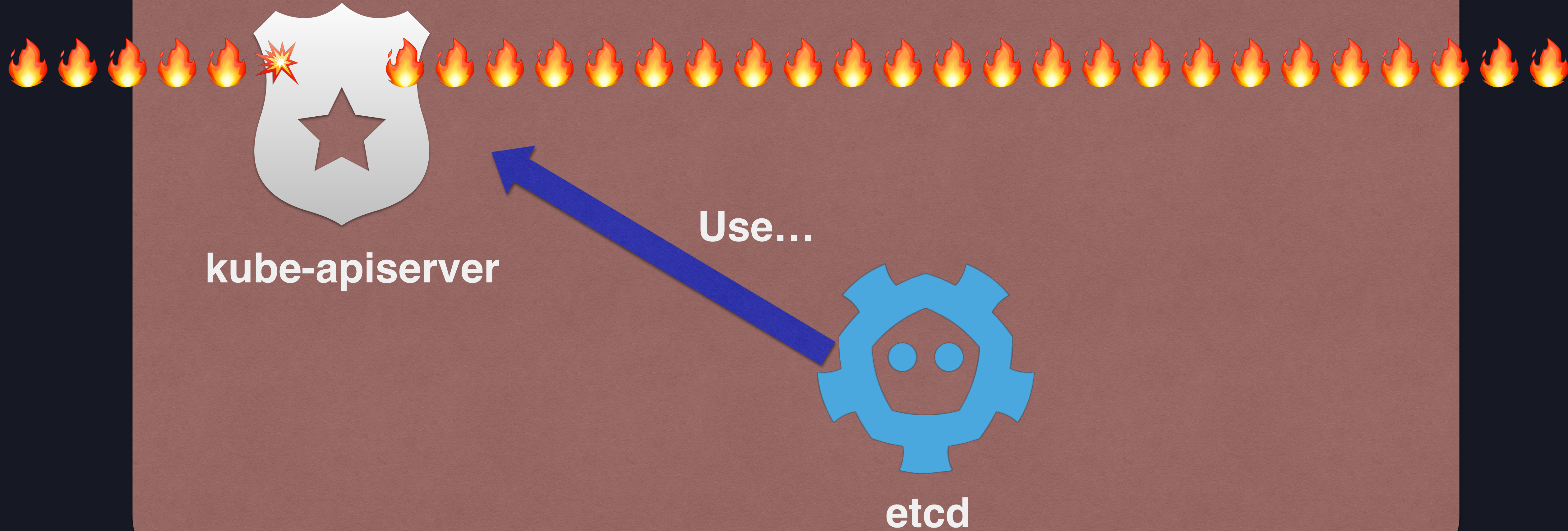
受け取った要求
に対応する操作
に対して、使用
するアカウント
が権限があるか。

Admission Control



AUTH JOURNEY

Authorization(for example: RBAC)



AUTH JOURNEY

Authorization(for example: RBAC)

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: hoge-ns
  name: service-reader-role
rules:
- apiGroups: [""]
  verbs: ["get", "list"]
  resources: ["services"]
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: service-reader-rolebinding
  namespace: hoge-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: service-reader-role
subjects:
- kind: ServiceAccount
  name: default
  namespace: hoge-ns
```



AUTH JOURNEY

Authentication

接続元が確かに用意されたアカウントのクライアントかどうか



Authorization

受け取った要求に対応する操作に対して、使用するアカウントが権限があるか。



Admission Control

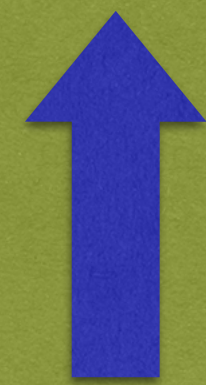
受け取った要求がクラスターリソースに課せられた制限内かどうか。



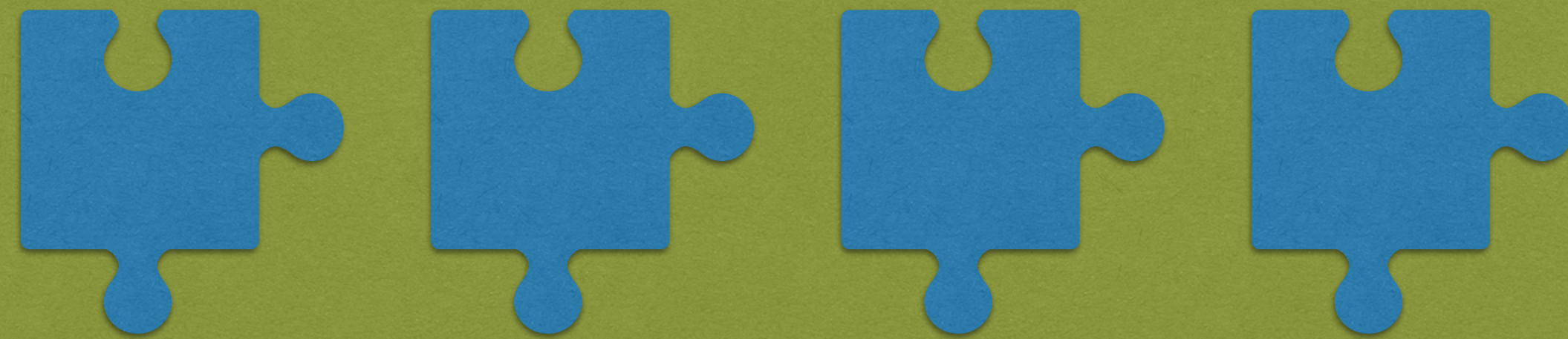
AUTH JOURNEY

Admission Controll

kube-apiserver

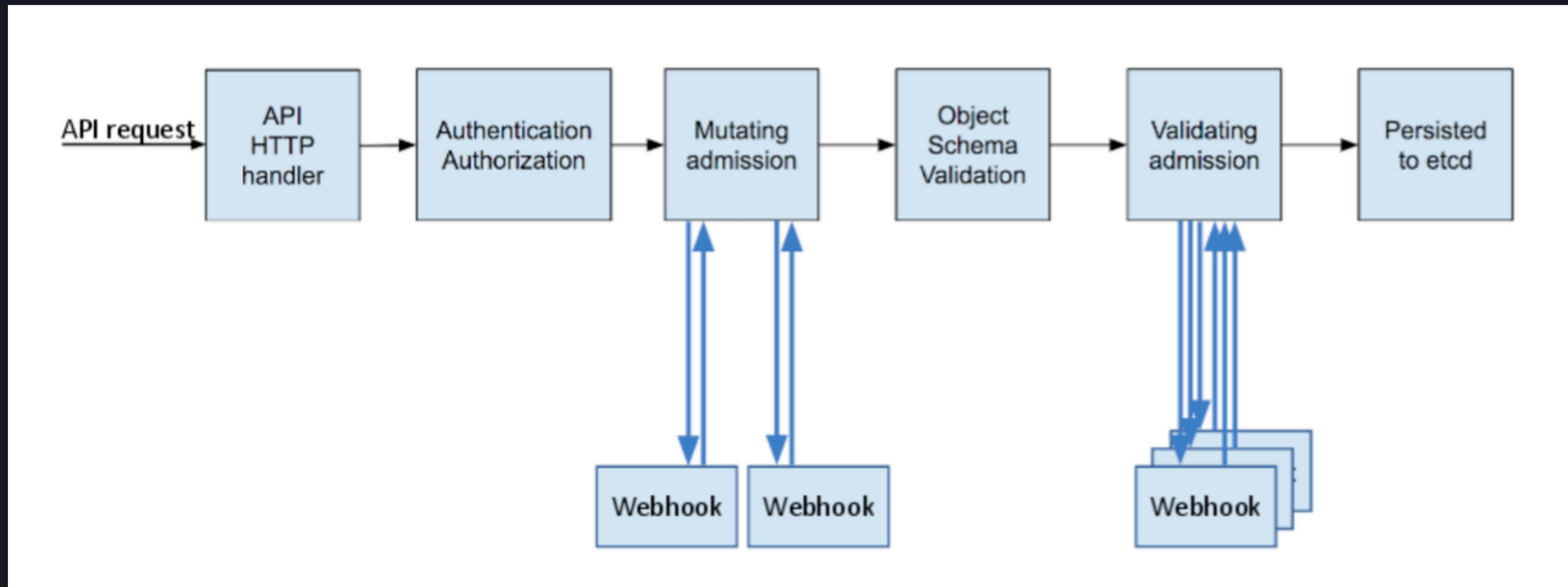


Use...



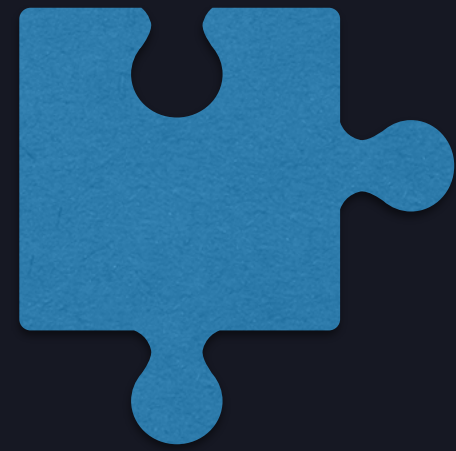
AUTH JOURNEY

Describe Admission Controll

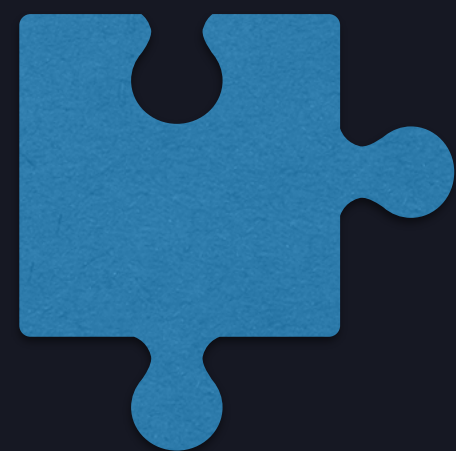


AUTH JOURNEY

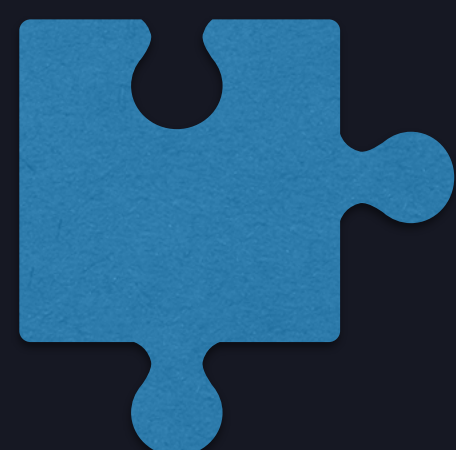
Admission Control Plugin Example



AlwaysAdmit...Accept All Request



SecurityContextDeny...Deny Security Context



AlwaysDeny...Deny All request



AUTH JOURNEY

Authentication

接続元が確かに用意されたアカウントのクライアントかどうか



Authorization

受け取った要求に対応する操作に対して、使用するアカウントが権限があるか。

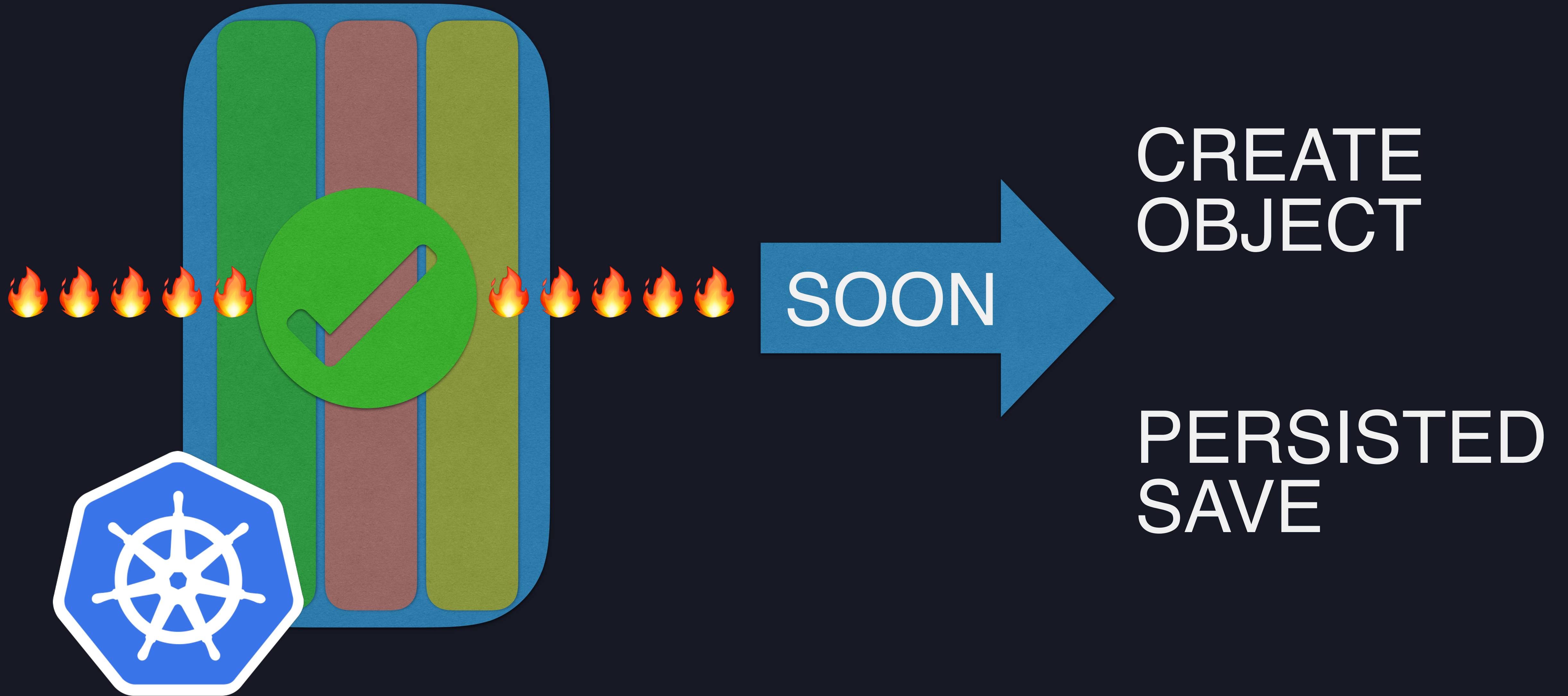


Admission Control

受け取った要求がクラスターリソースに課せられた制限内かどうか。



AUTH JOURNEY



AUTH JOURNEY

kube-apiserver

Request

```
/apps/v1beta2/deployment
```



```
/apps/v1/namespace  
/apps/v1/configmap  
/apps/v1/service  
/apps/v1beta2/deployment  
⋮
```

```
* 「リソース  
登録ヲ確認  
シマシタ。」
```



AUTH JOURNEY

Request

`/apps/v1beta2/deployment`



kube-apiserver



FORM

JSON



REQUEST
VALIDATION



AUTH JOURNEY



JSON DESELIZE



VALIDATION
REQUEST FORM



AUTH JOURNEY

Request

`/apps/v1beta2/deployment`

kube-apiserver



REQUEST
VALIDATION



AUTH JOURNEY



AUTH JOURNEY



ETCD

ETCD IS...

1. USABLE HTTP WITH JSON
2. SECURE TLS ENCRYPT
3. RAPID KVS (FOCUS READ)
4. DISTRIBUTED
5. BACKEND BBOLT



AUTH JOURNY



BBOLT

BBOLT IS...

1. ETCD BACKEND

2. FULLSERIAL TRANSACTION

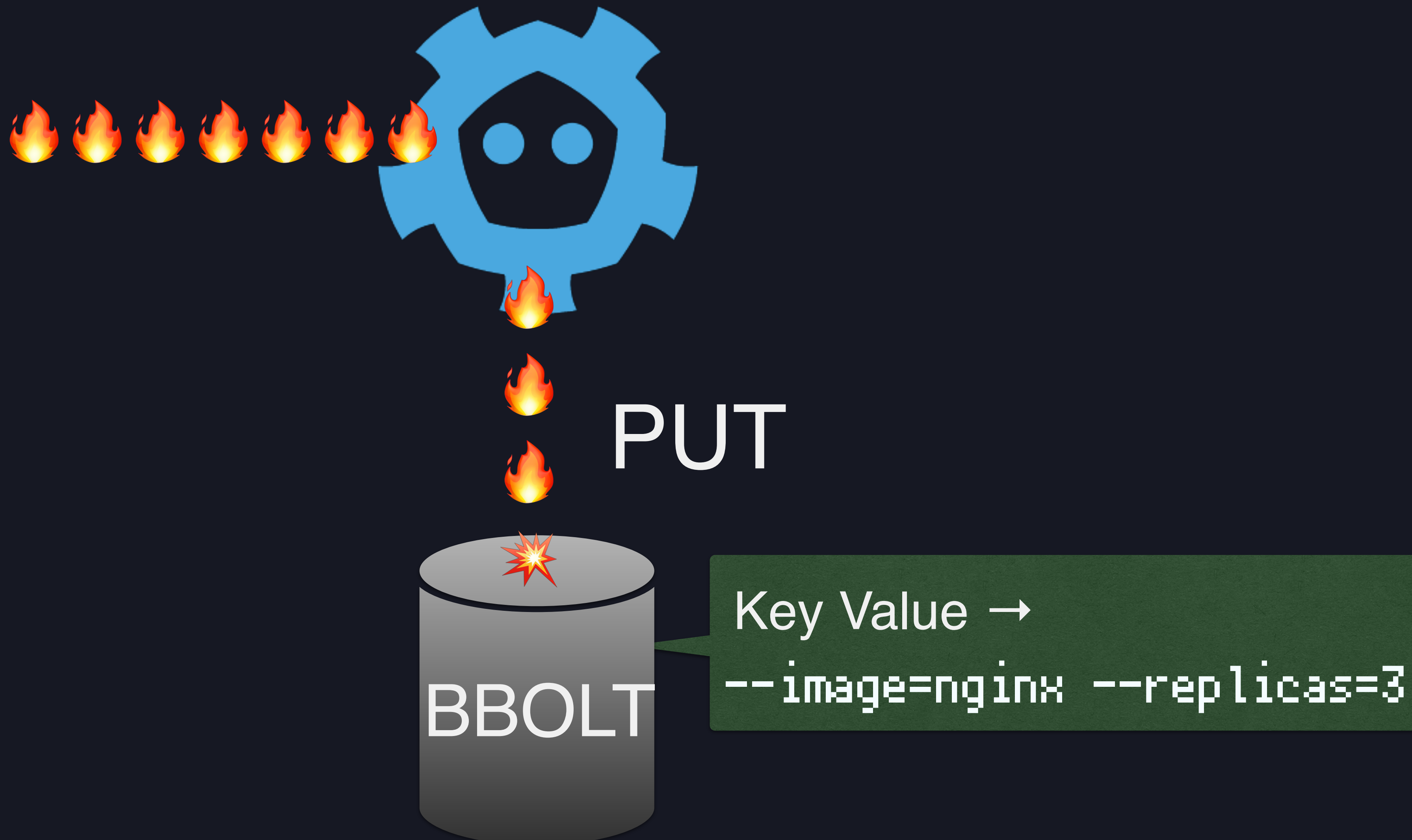
3. ACID SEMANTICS

4. LOCK FREE

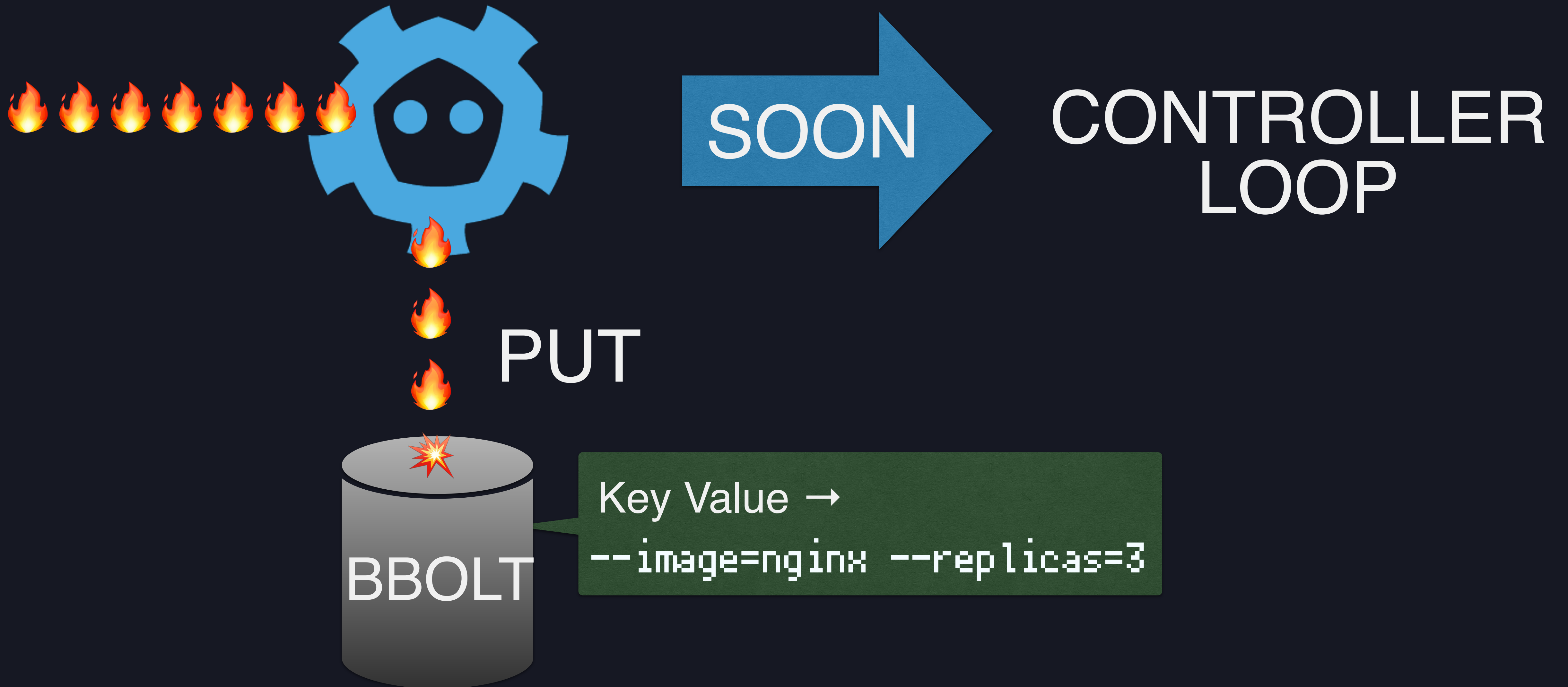
5. SINGLE WRITE MULTI READ



AUTH JOURNEY



AUTH JOURNEY



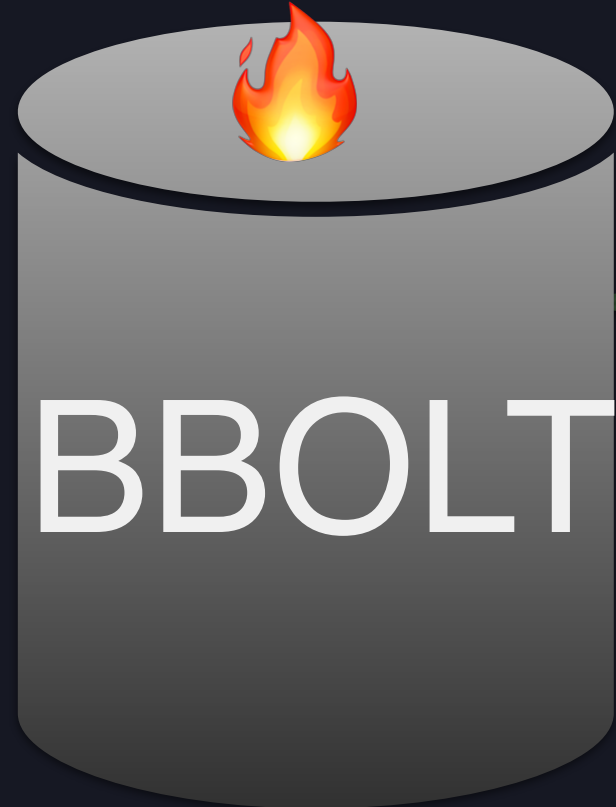
NEXT...



CONTROLLER LOOP



GET



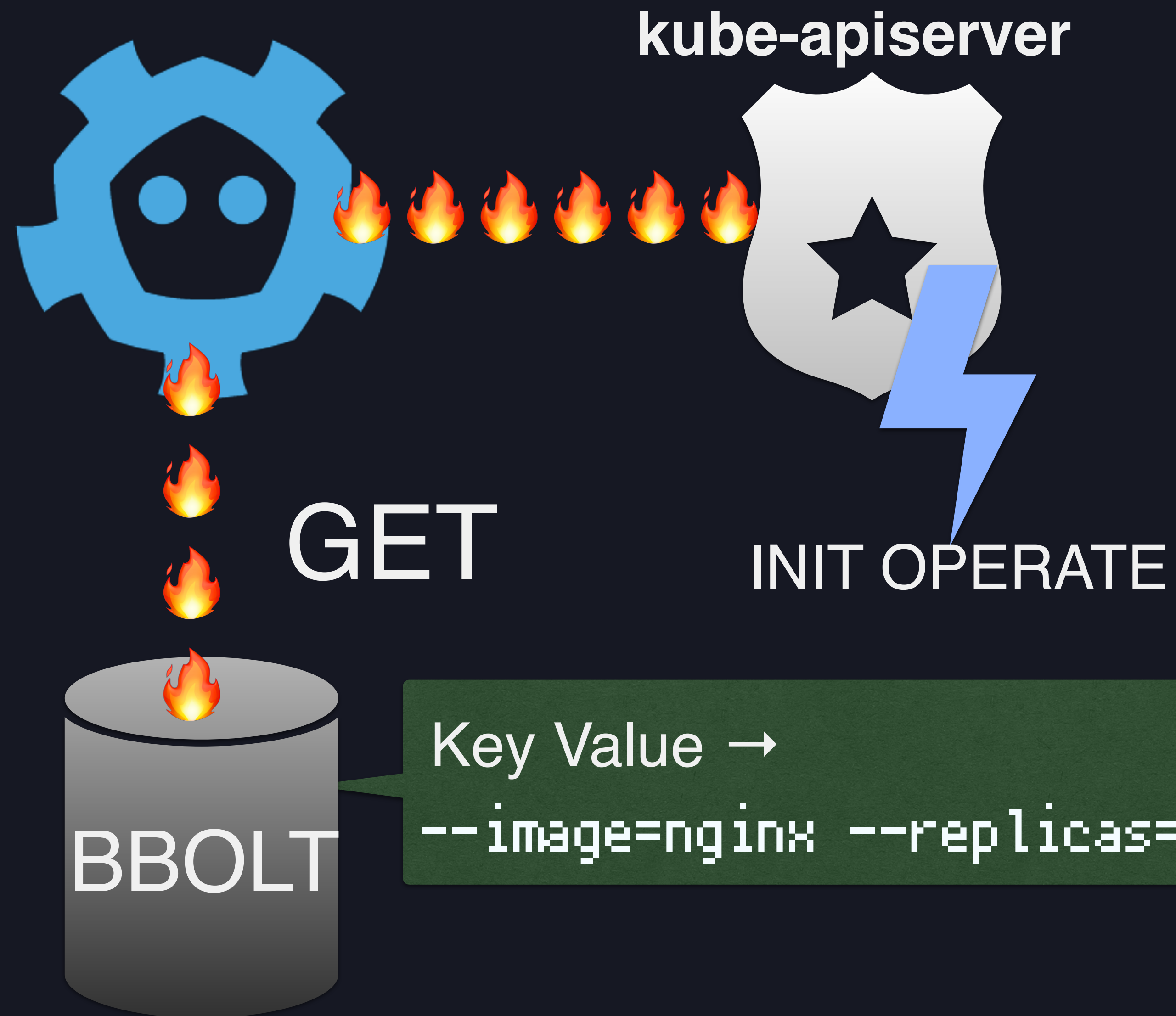
```
Key Value →  
--image=nginx --replicas=3
```

詠唱中



BEFORE
GET
RESOURCE..

CONTROLLER LOOP



詠唱中



BEFORE
GET
RESOURCE..



CONTROLLER LOOP

INIT OPERATE IS...



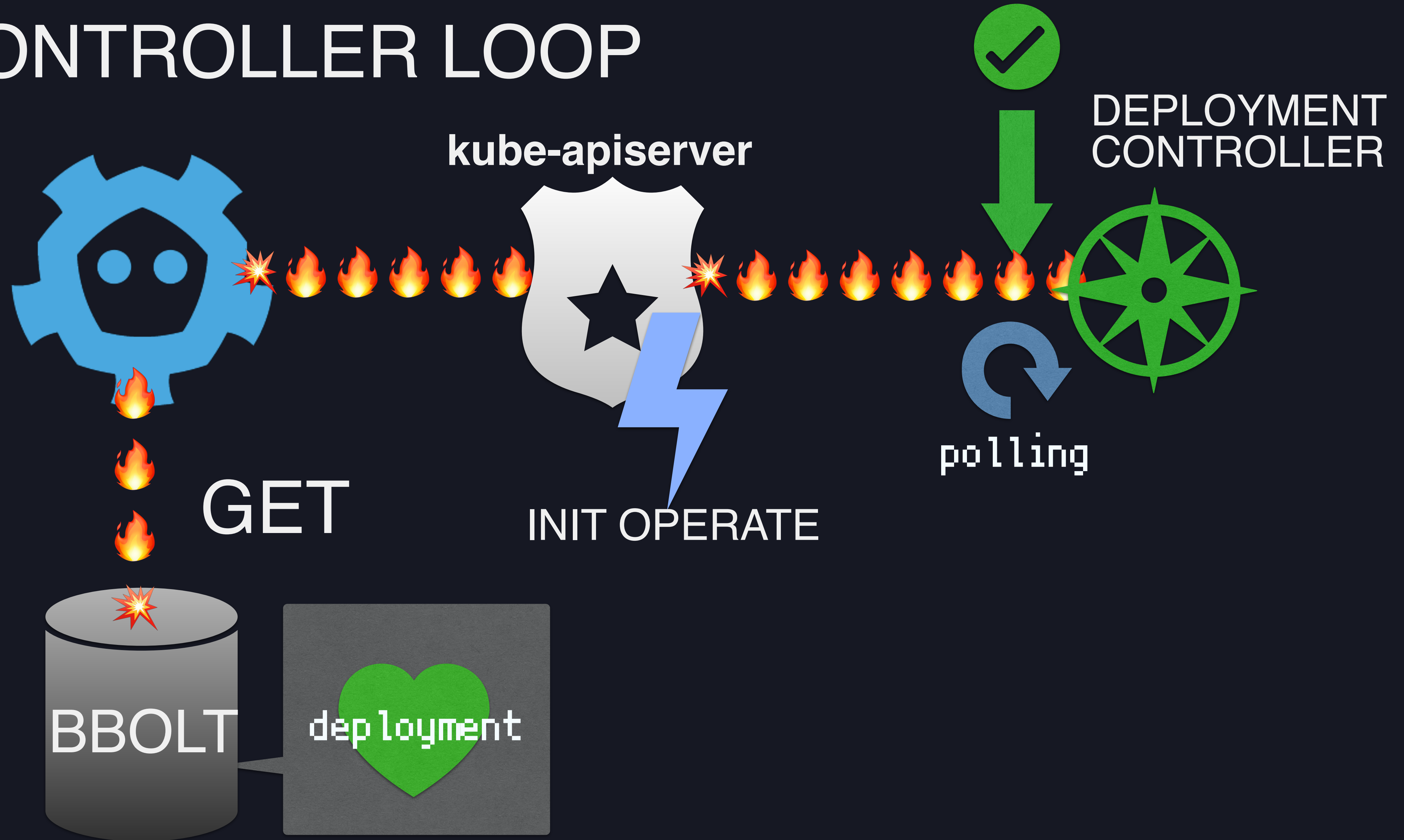
INITIALIZERS

SETUP
EXPEIMENSIBLE RESOURCE

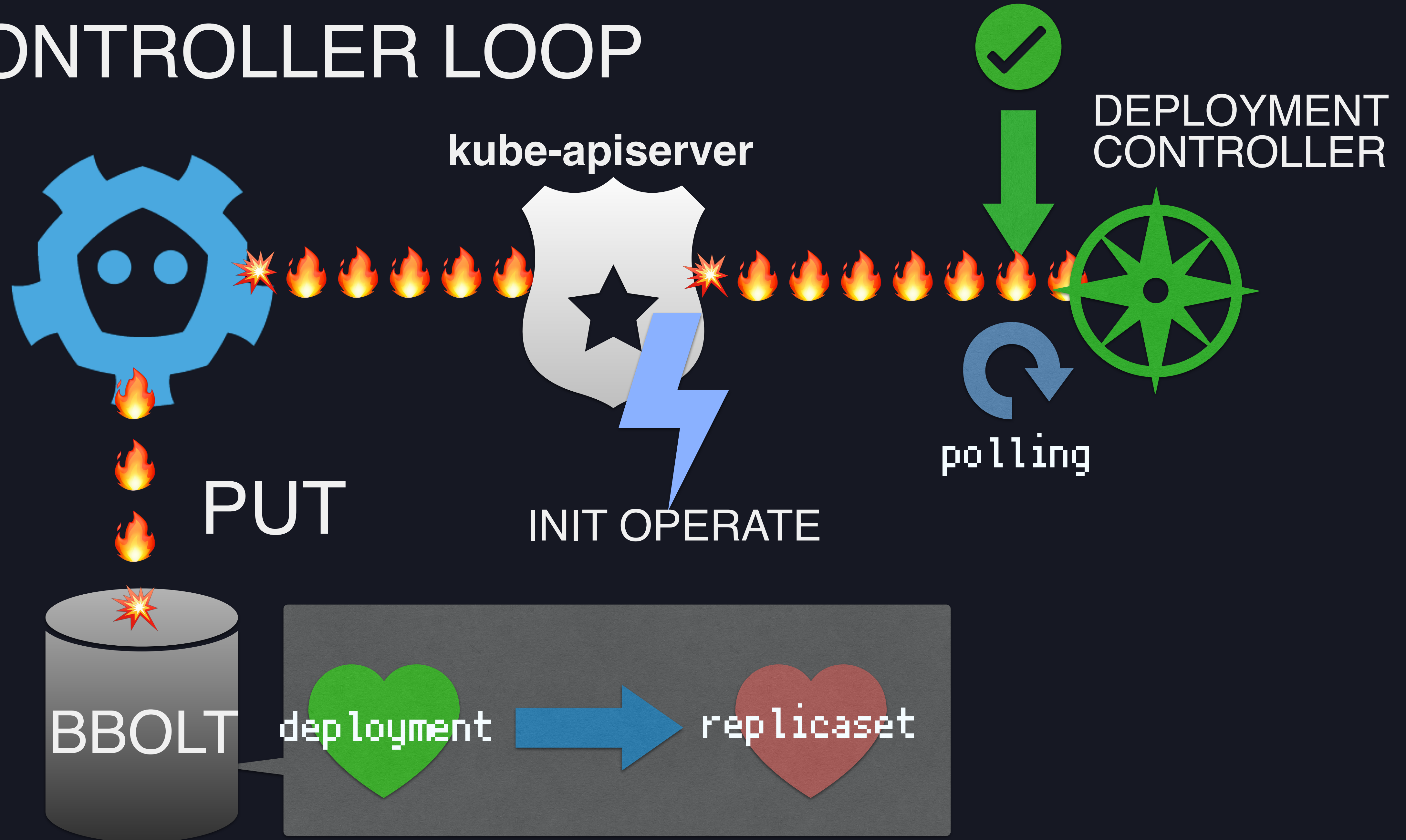
- INSERT PROXY SIDECAR
- TOO LONG PASSWORD IN SECRET
- SEE <https://ahmet.im/blog/initializers/>



CONTROLLER LOOP



CONTROLLER LOOP



CONTROLLER LOOP

DEPLOYMENT
CONTROLLER



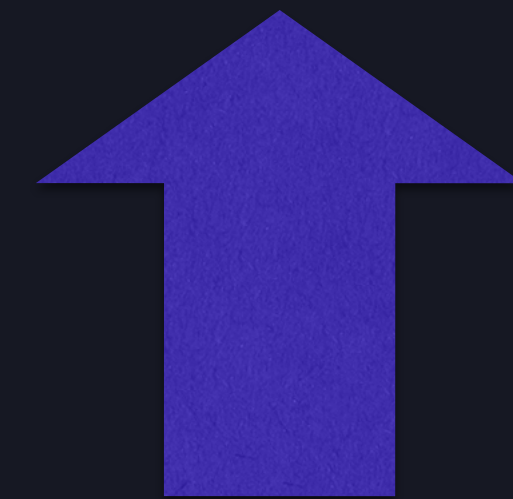
REPLICASET



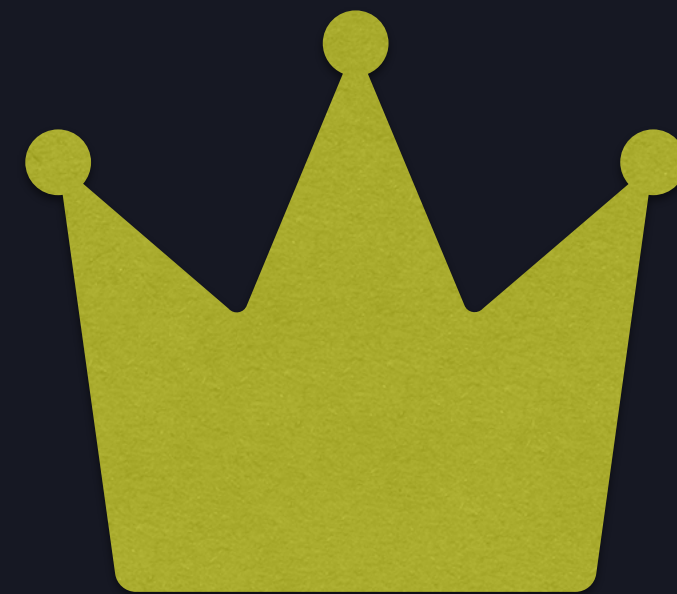
ENDPOINT
CONTROLLER



Service Account
& Token
CONTROLLER



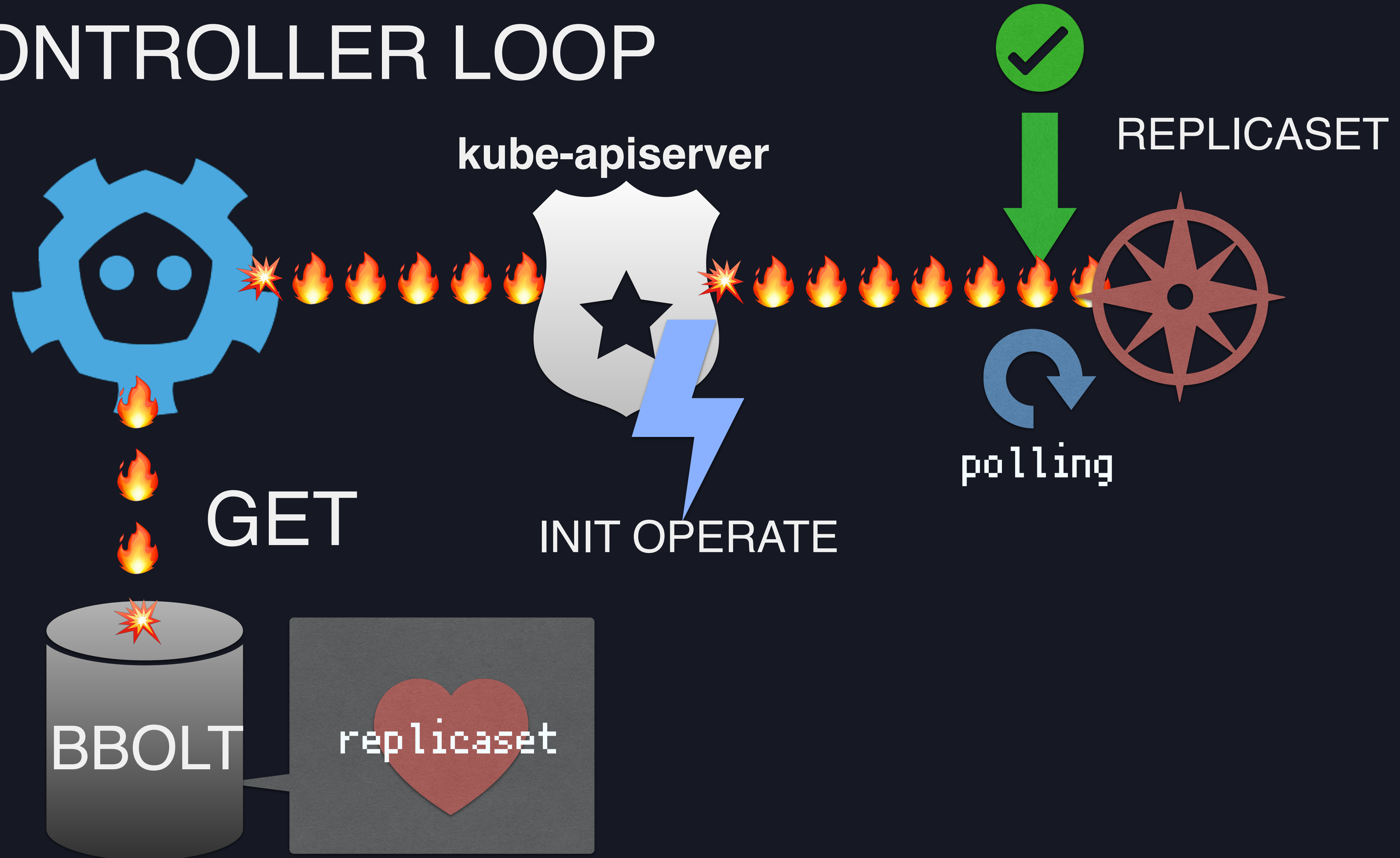
MANAGE



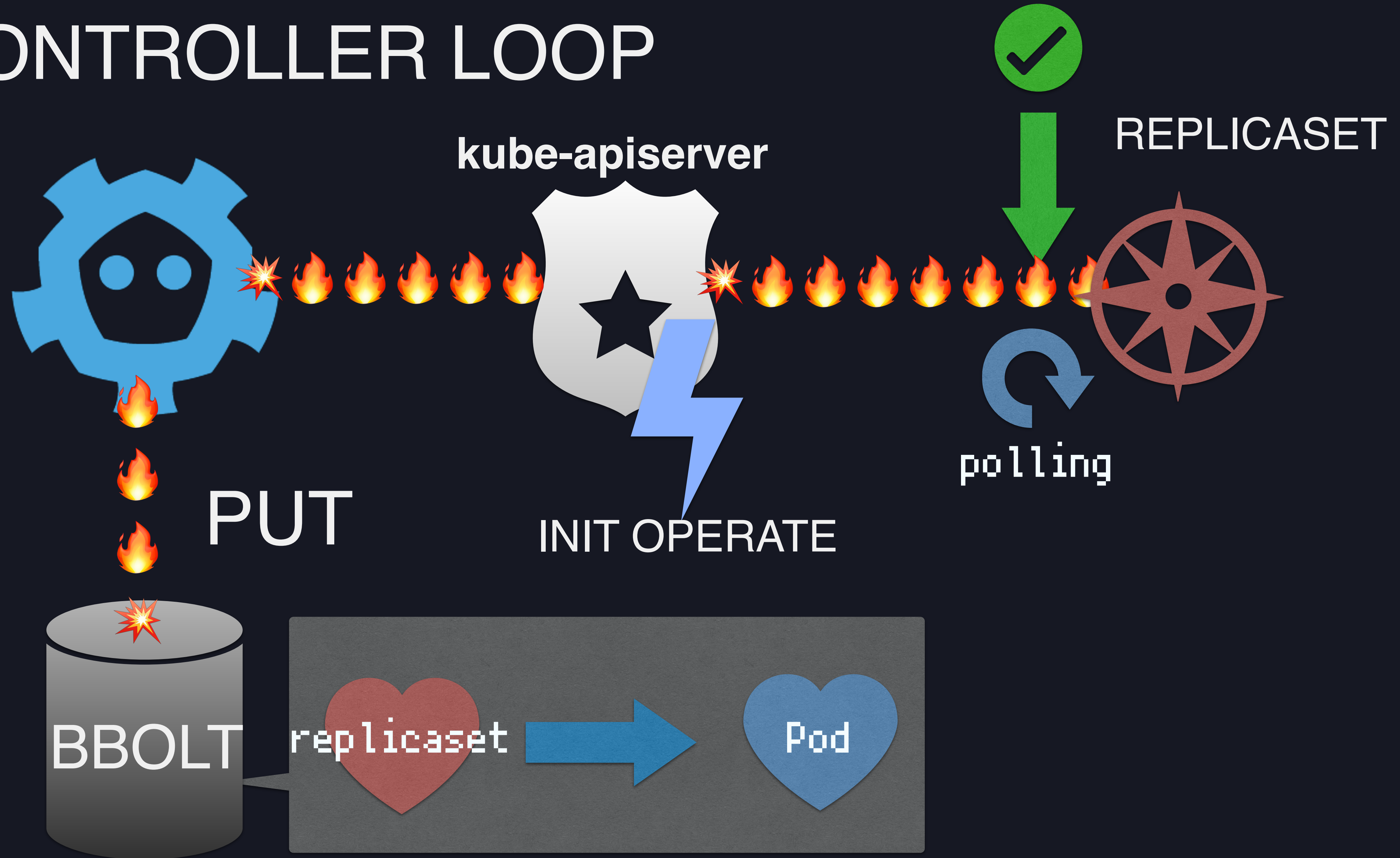
kube-controller-manager



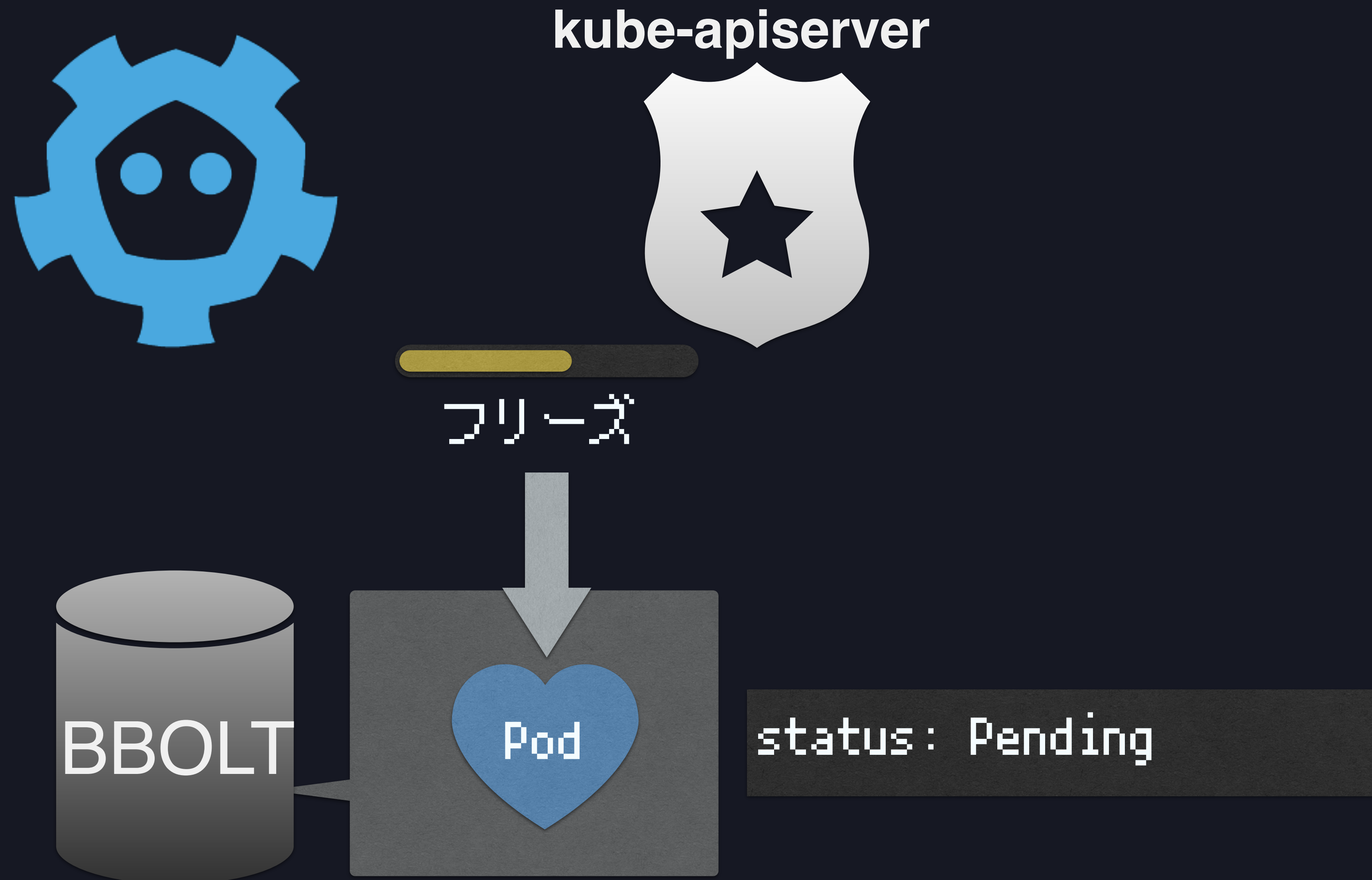
CONTROLLER LOOP



CONTROLLER LOOP



CONTROLLER LOOP



CONTROLLER LOOP

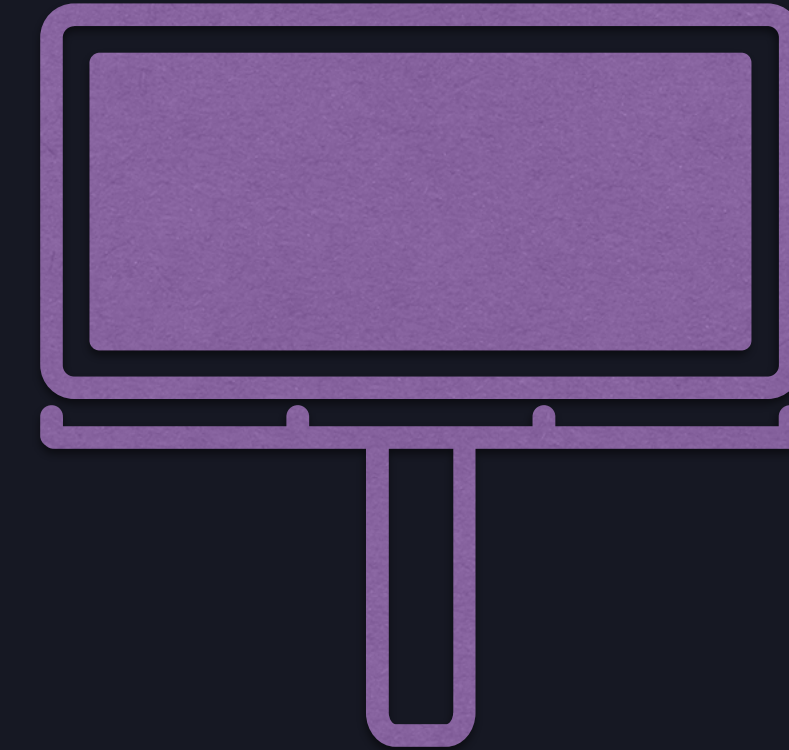


kube-apiserver

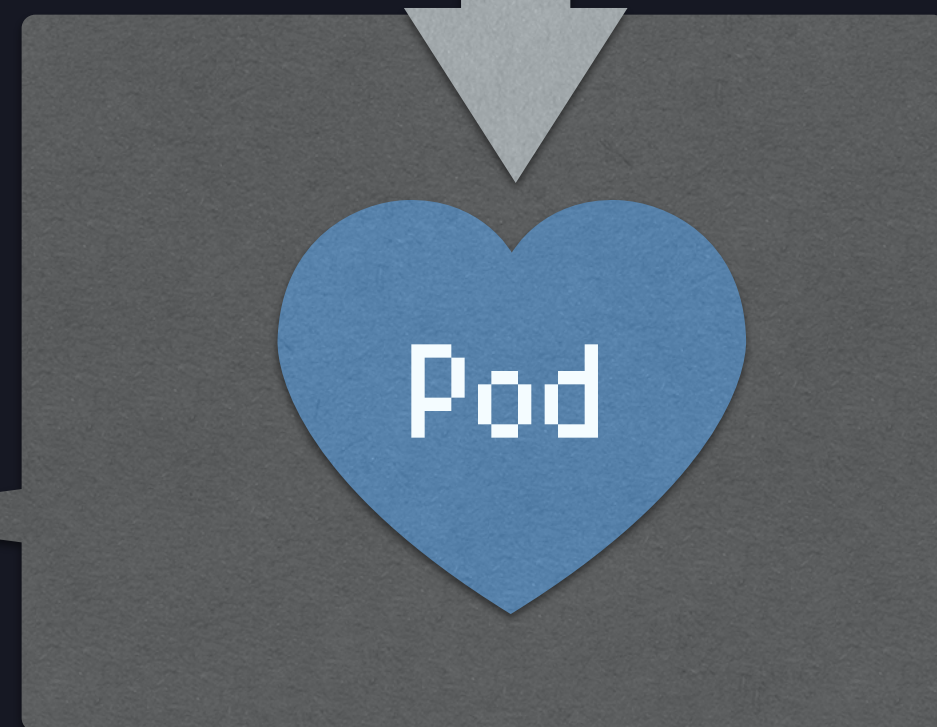
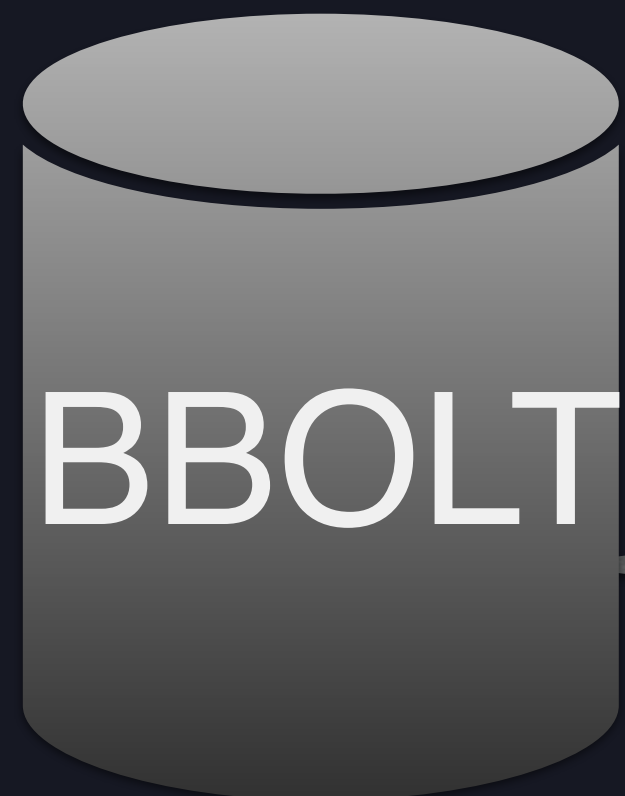


詠唱中

kube-scheduler



リリース



status: Pending



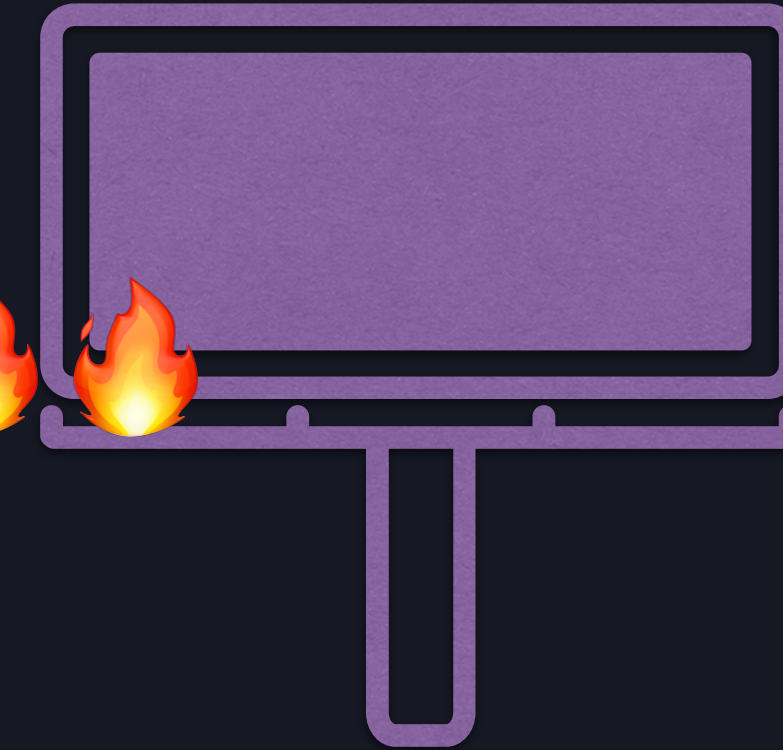
CONTROLLER LOOP



kube-apiserver

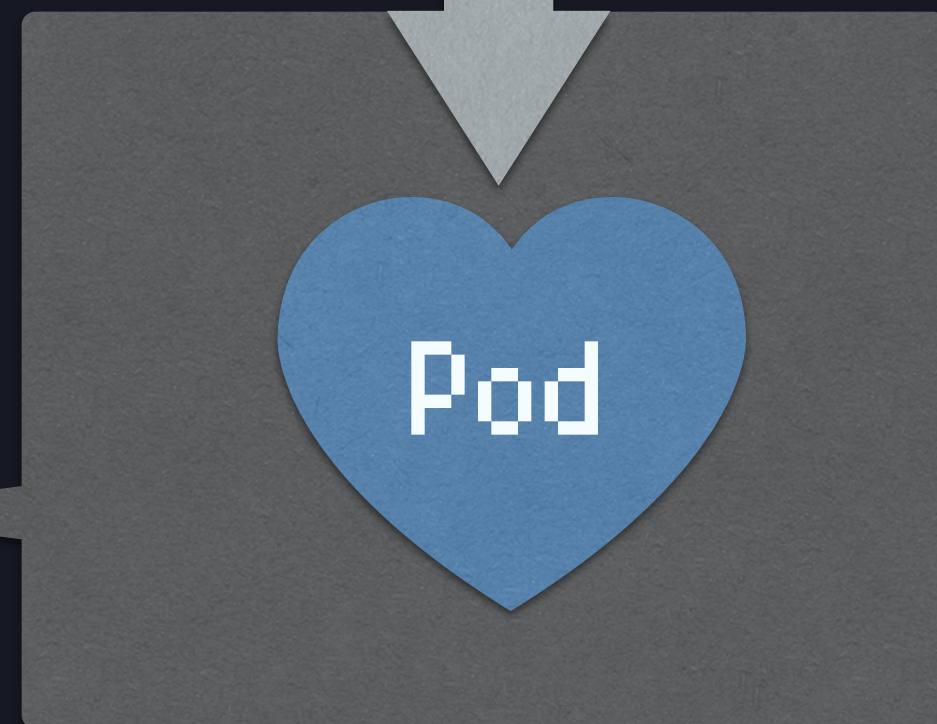


kube-scheduler



GET

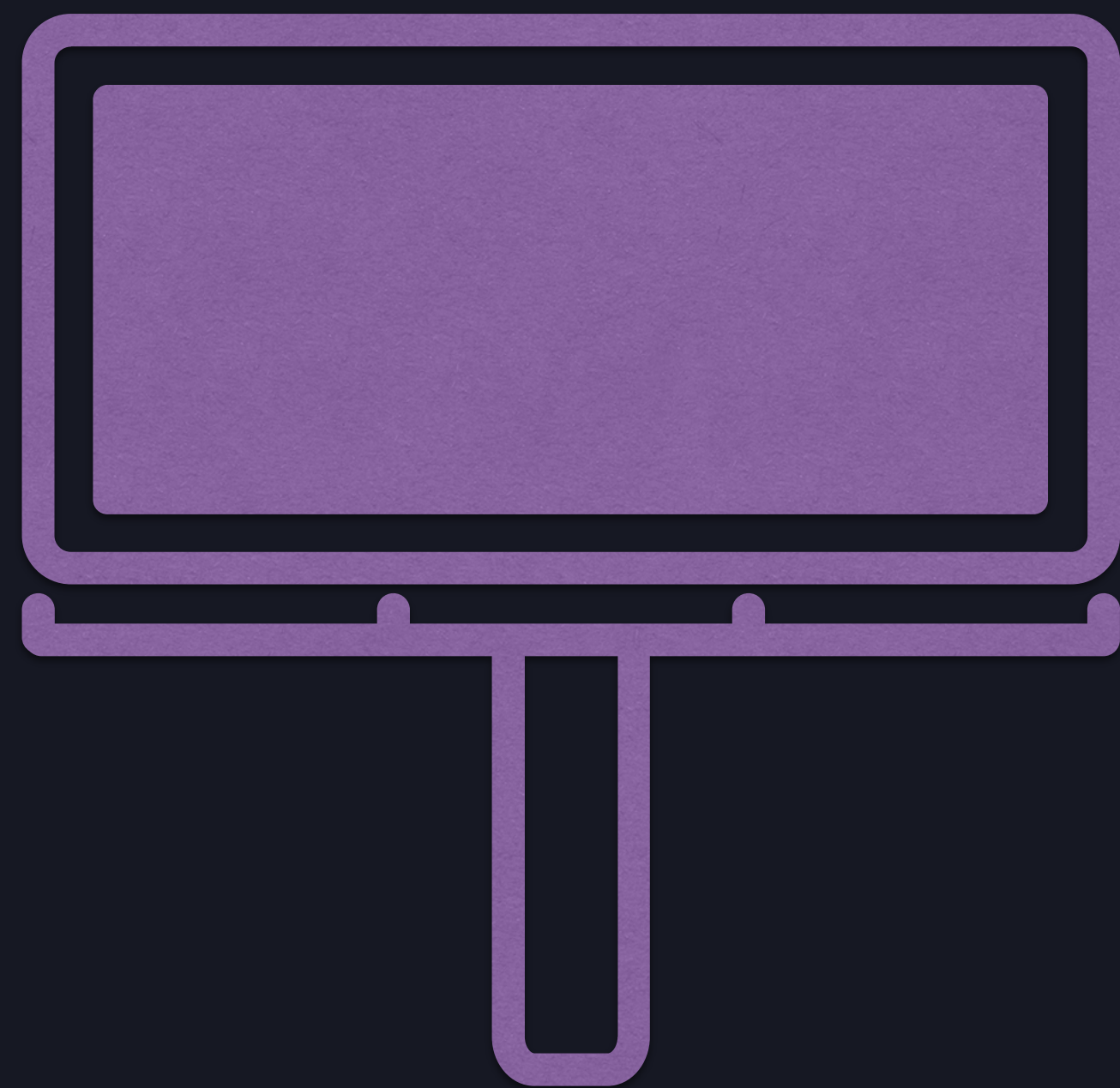
詠唱中



```
status: Pending
```



CONTROLLER LOOP



kube-scheduler

KUBE-SCHEDULER OPERATE....

1. FILL PODSPACE NODENAME FOR EMPTY VALUE POD
2. CHECK NODE RESOURCE
3. BIND POD TO NODE



CONTROLLER LOOP

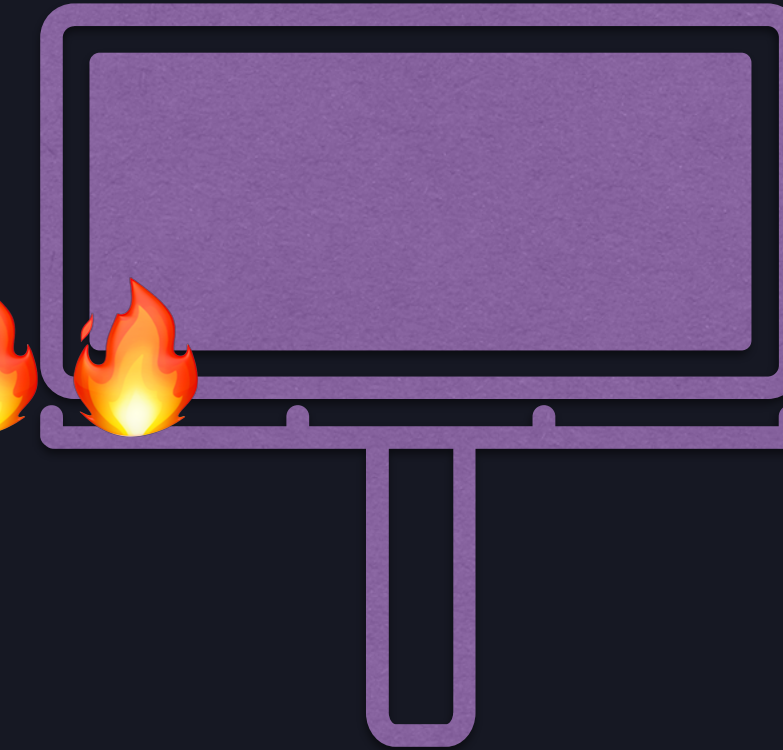


kube-apiserver



POST

kube-scheduler

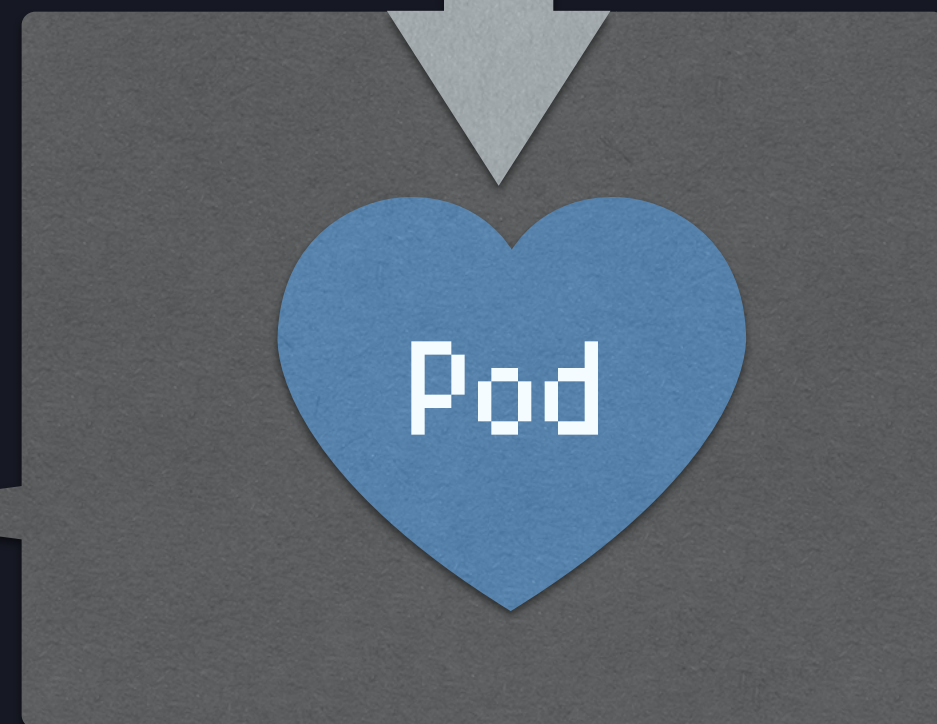


PUT

詠唱中



BBOLT



Pod

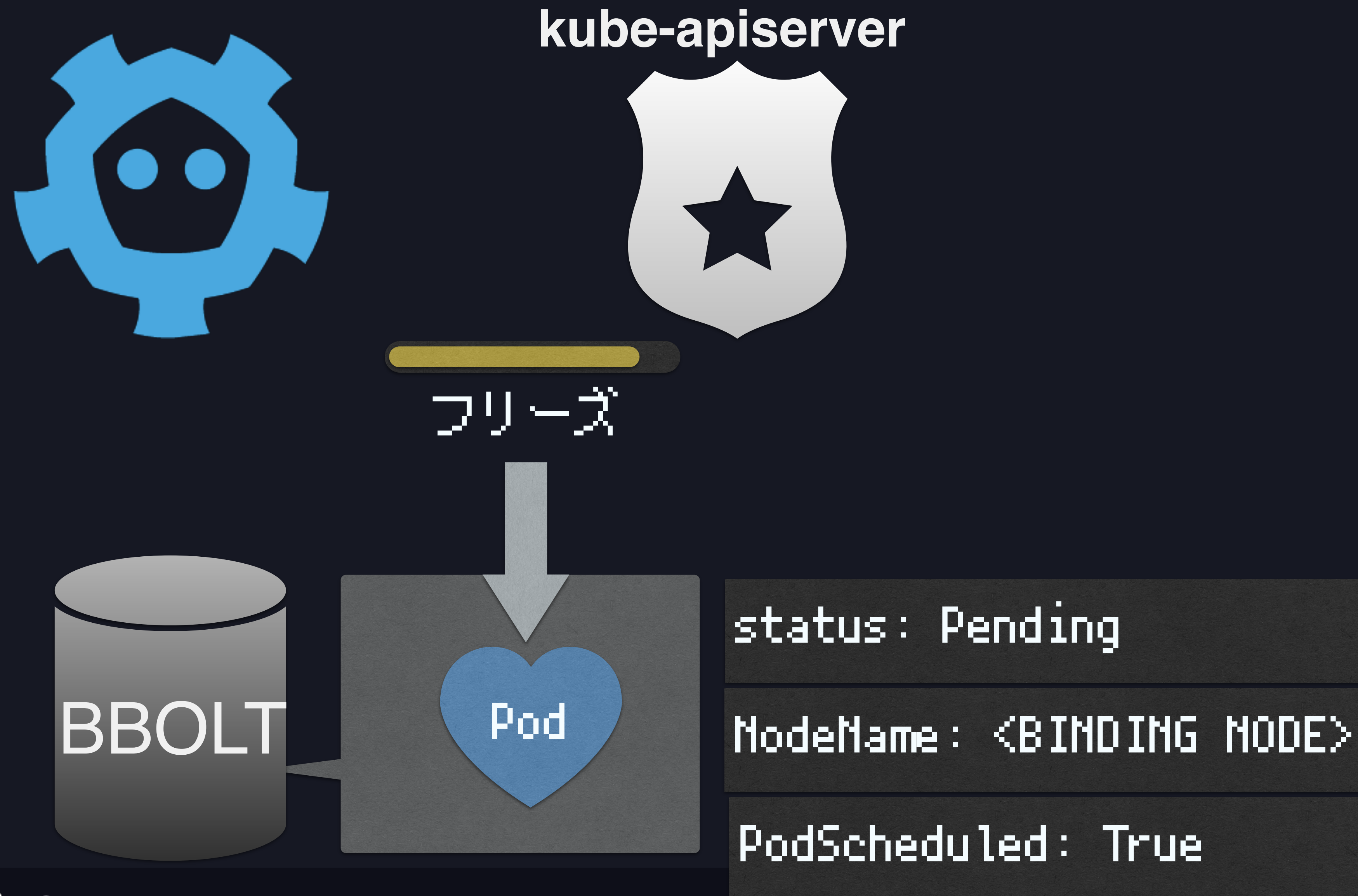
```
status: Pending
```

```
nodeName: <BINDING NODE>
```

```
podScheduled: True
```



CONTROLLER LOOP



CONTROLLER LOOP



kube-apiserver



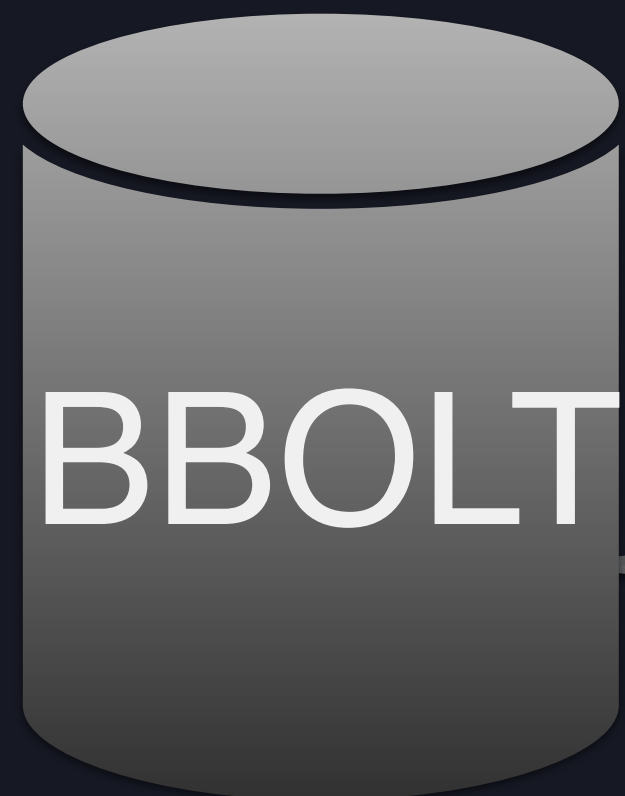
詠唱中



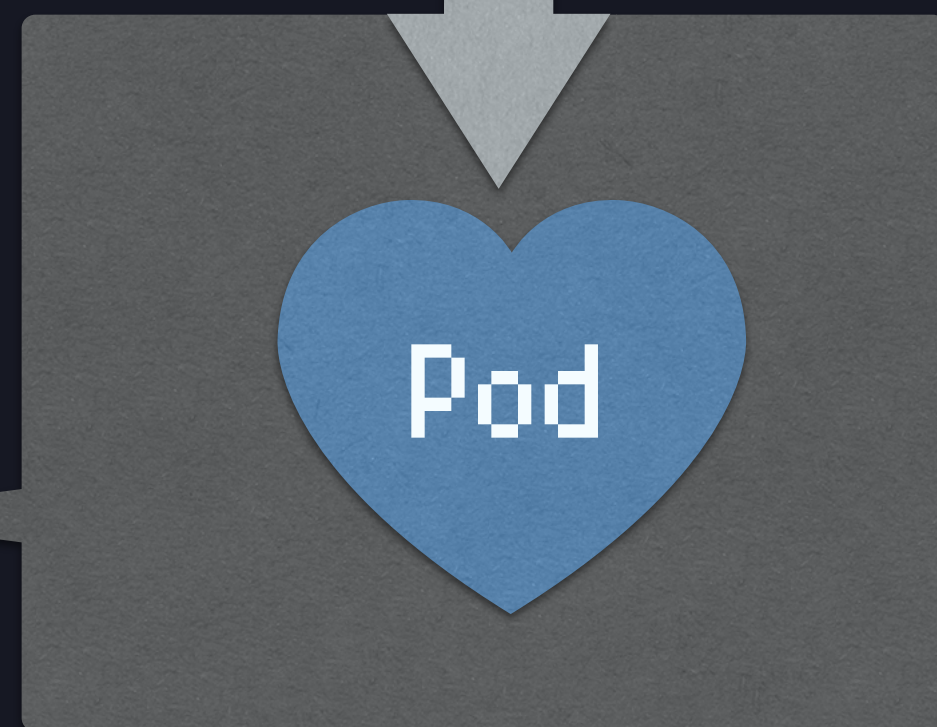
kubelet



リリース



BBOLT



```
status: Pending
```

```
nodeName: <BINDING NODE>
```

```
podScheduled: True
```



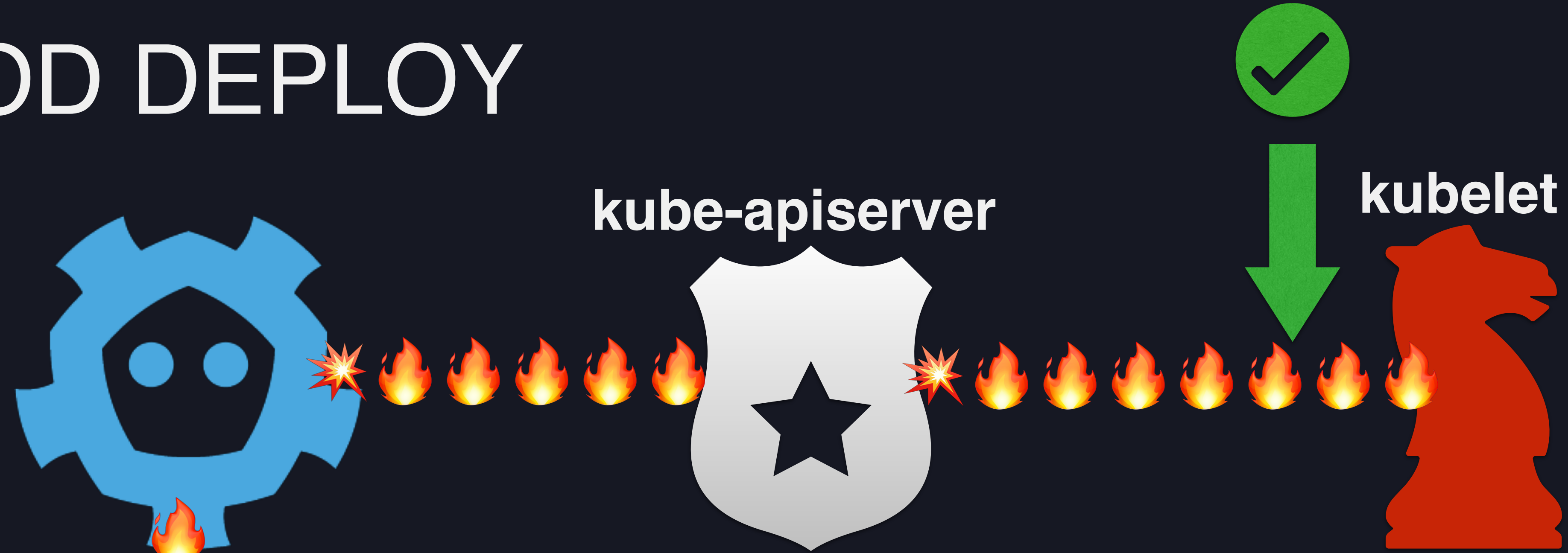
NEXT...



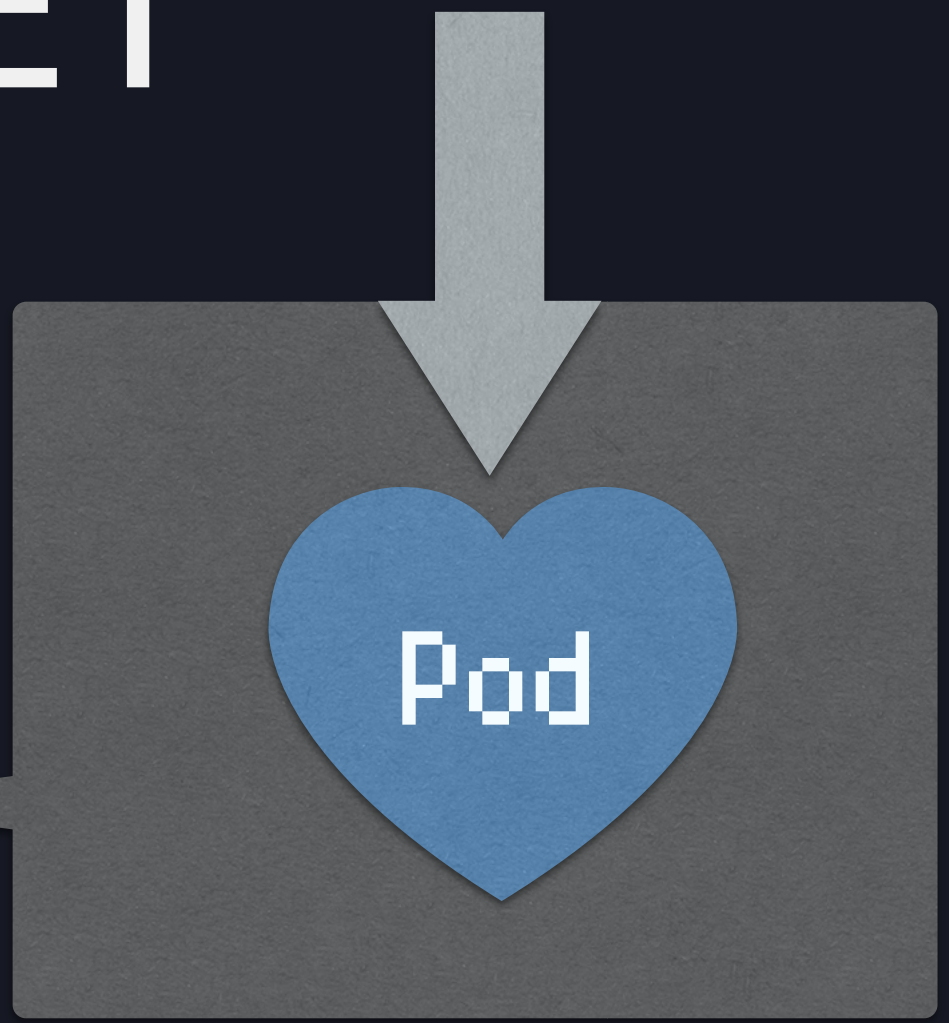
POD DEPLOY



POD DEPLOY



GET 詠唱中



```
status: Pending
nodeName: <BINDING NODE>
podScheduled: True
```

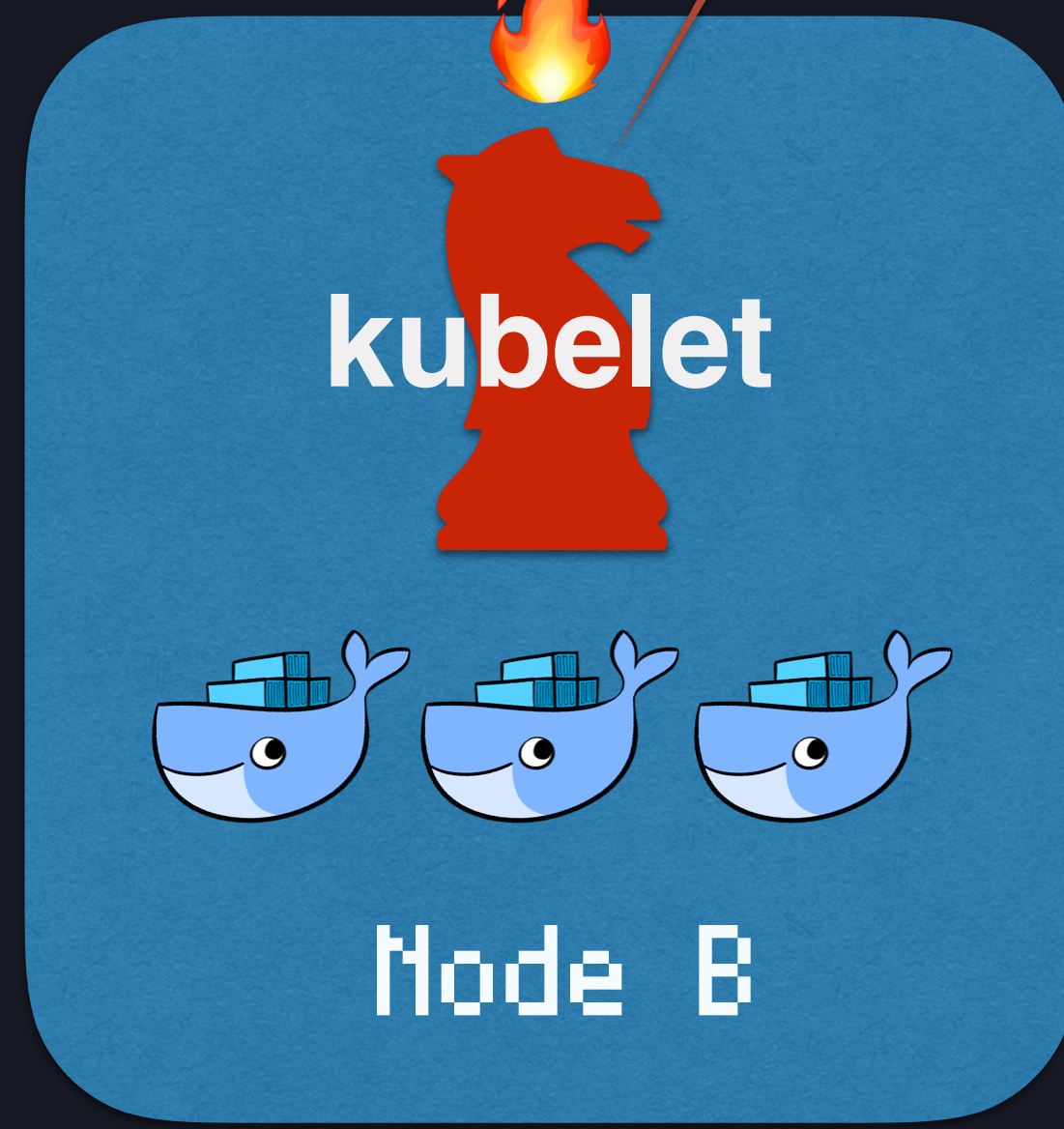
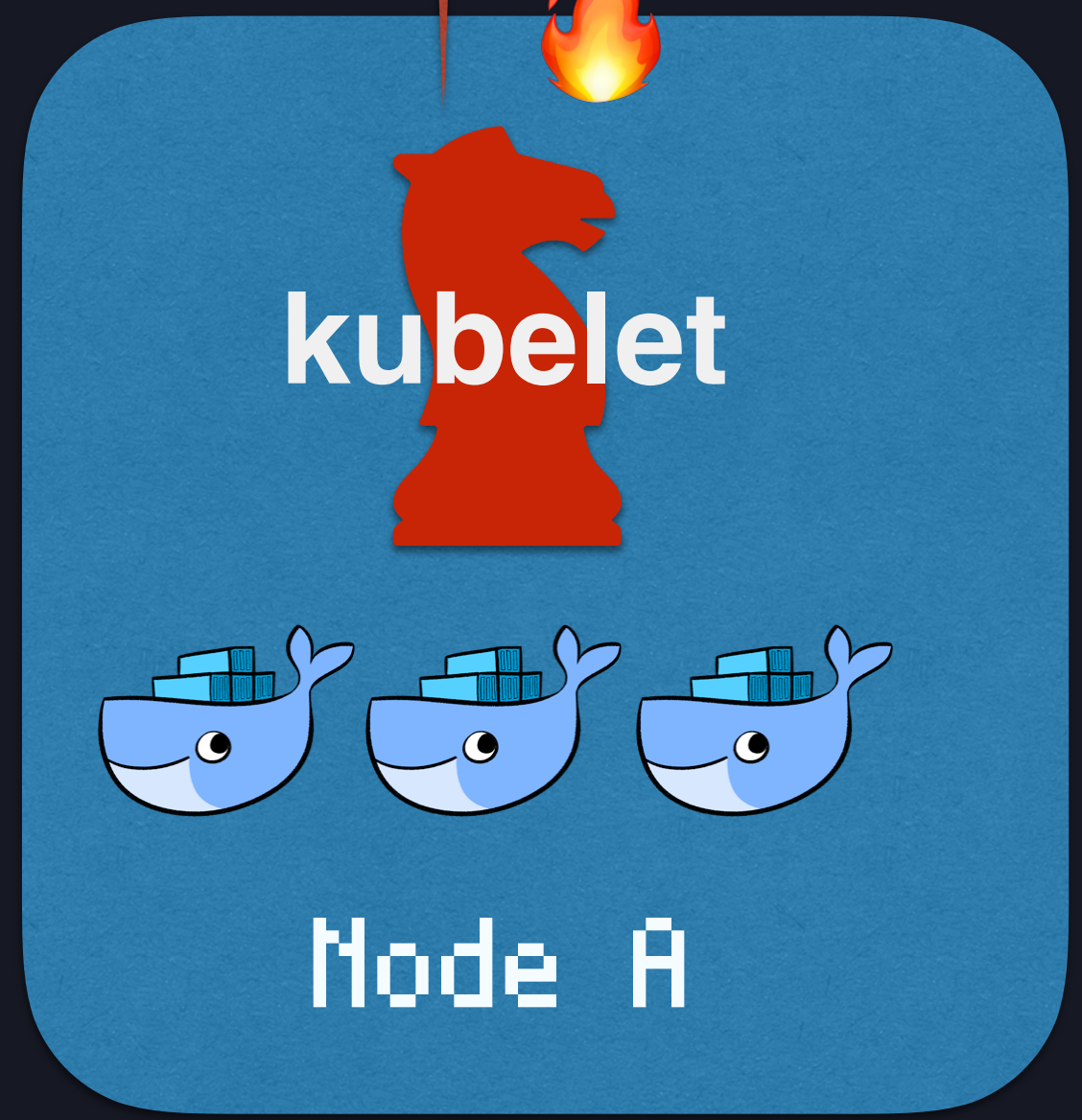
POD DEPLOY



Node AのPodの状態は?

Node BのPodの状態は?

Node CのPodの状態は?



POD DEPLOY



kubelet

KUBELET OPERATE....

1. SYNC POD STATUS
IN ETCD AND LOCAL CACHE
2. CREATE CGROUP
3. BIND POD AND VOLUME
4. BIND POD AND SECRET



POD DEPLOY



kubelet

CONTAINER

VOLUMES

POD METADATA

x N



POD DEPLOY



kubelet

**KUBELET OPERATE...
(CASE DOCKER)**

5.CREATE PAUSE CONTAINER



POD DEPLOY



kubelet

CONTAINER

BASE PAUSE IMAGE

VOLUMES

POD METADATA

x N



POD DEPLOY



kubelet

**KUBELET OPERATE....
(CASE DOCKER)**

6. ATTACHE NETWORK IF



POD DEPLOY



kubelet



x N



POD DEPLOY



kubelet

**KUBELET OPERATE...
(CASE DOCKER)**

7. PULL CONTAINER IMAGE

8. RUN CONTAINER IMAGE 🎉



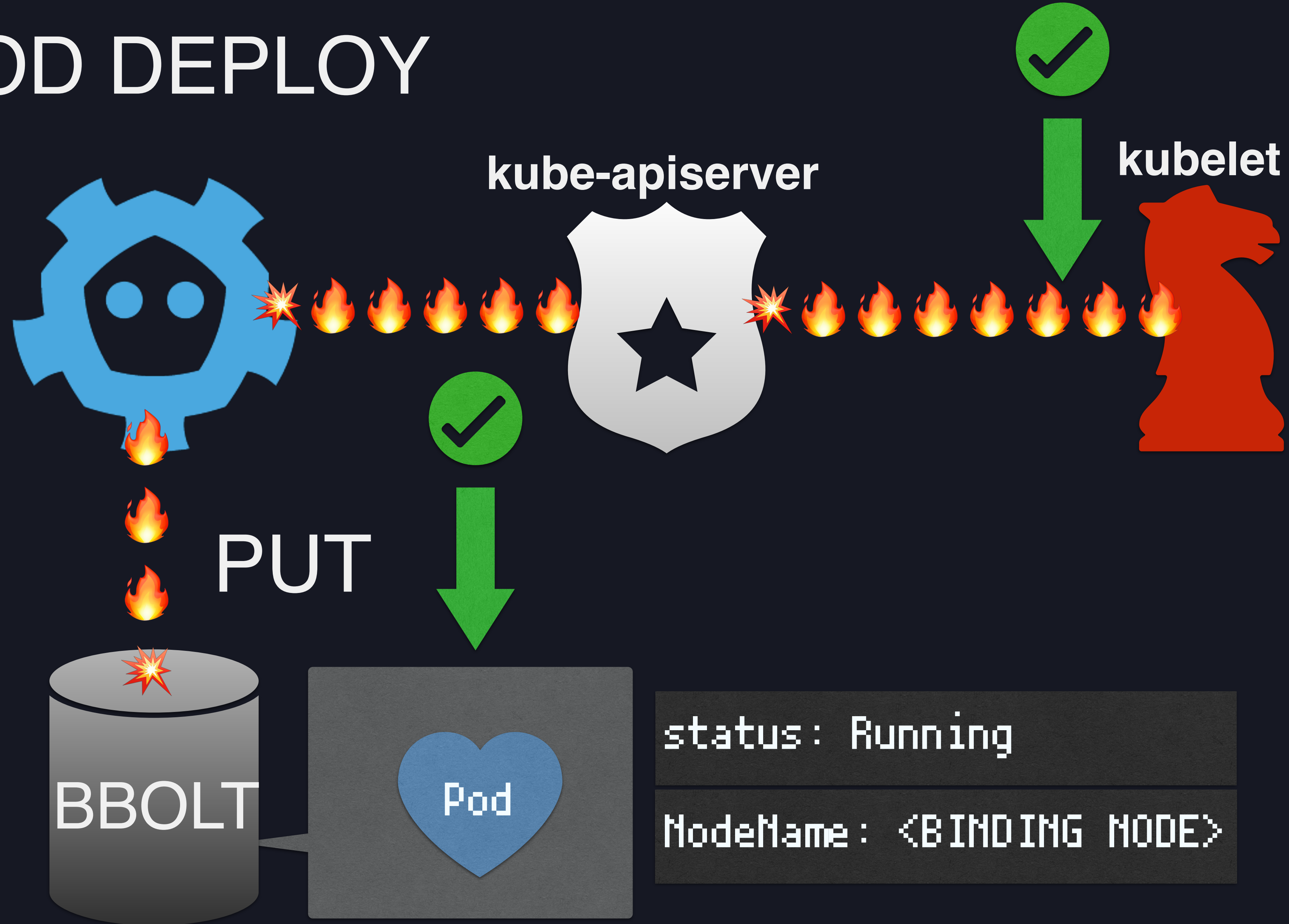
POD DEPLOY



kubelet



POD DEPLOY



POD DEPLOY



たたかう

👉 `kubectl run --image=nginx --replicas=3`

`kubectl get pod --all-namespaces`

`sudo rm -rf / --preserve-root`

コンテナをやめる



POD DEPLOY



たたかう

```
kubectl run --image=nginx --replicas=3
```

👉 `kubectl get pod --all-namespaces`

```
sudo rm -rf / --preserve-root
```

コンテナをやめる



POD DEPLOY



たたかう

```
kubectl run --image=nginx --replicas=3
```

👉 `kubectl get pod --all-namespaces`

```
sudo rm -rf / --preserve-root
```

コンテナをやめる

```
$ kubectl get pod --all-namespaces
NAME                                READY   STATUS    RESTARTS   AGE
...
nginx-65cf545976-22nsz             1/1     Running   0          2d
nginx-65cf545976-21ljh             1/1     Running   0          2d
nginx-65cf545976-21ljh             1/1     Running   0          2d
...
```



FIN

THANKS 🙌

いらすとやの『中二病の女の子』のイラストが
好きすぎてファンアートを描いてしまった。By @Aiuti01

<https://twitter.com/Aiuti01>

<https://togetter.com/li/1221674>



CNDJP

