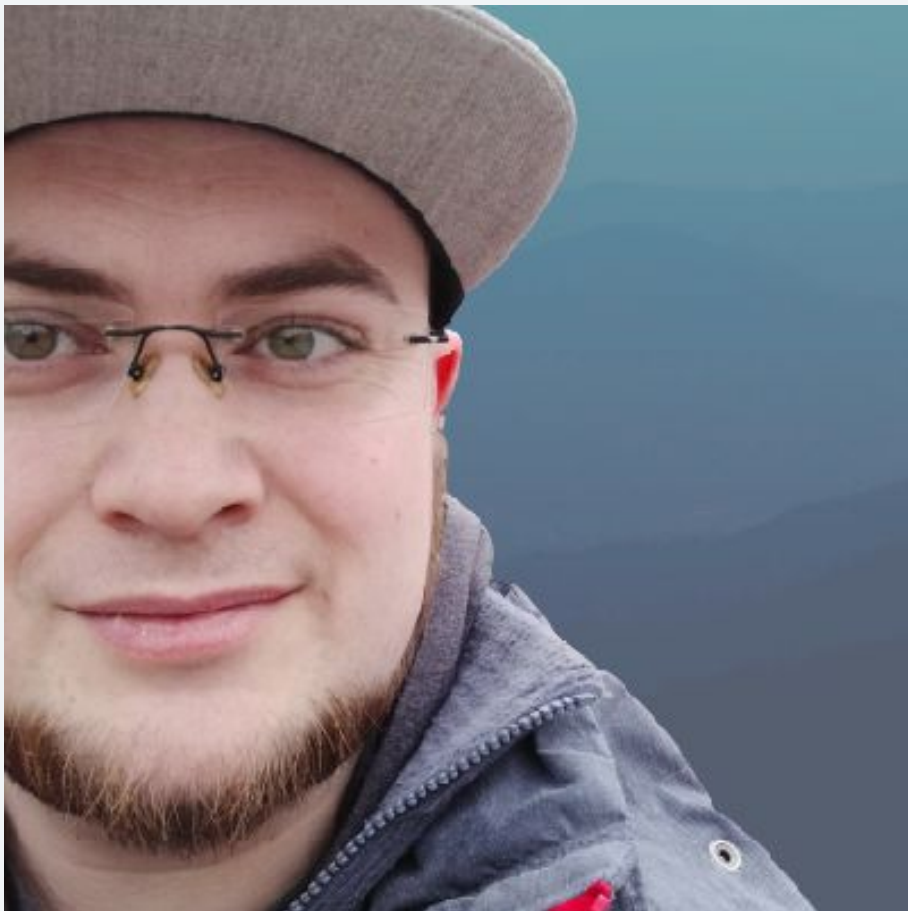


Managing Kubernetes without losing your cool


DevOps Notts
29th March 2022

KUBERNETES



Hi 🙋,

I'm **Marcus Noble**, a *platform engineer* at  **Giant Swarm** We're hiring

I'm found around the web as **✨AverageMarcus✨** in most places and **@Marcus_Noble_** on Twitter 

I have ~5 years experience running Kubernetes in production environments.

I also like home automation, IOT and 3D printing.



Summary

My 10 tips for working with Kubernetes

#1 → #5

Anyone can start using these today

#6 → #7

Good to know a little old-skool ops first

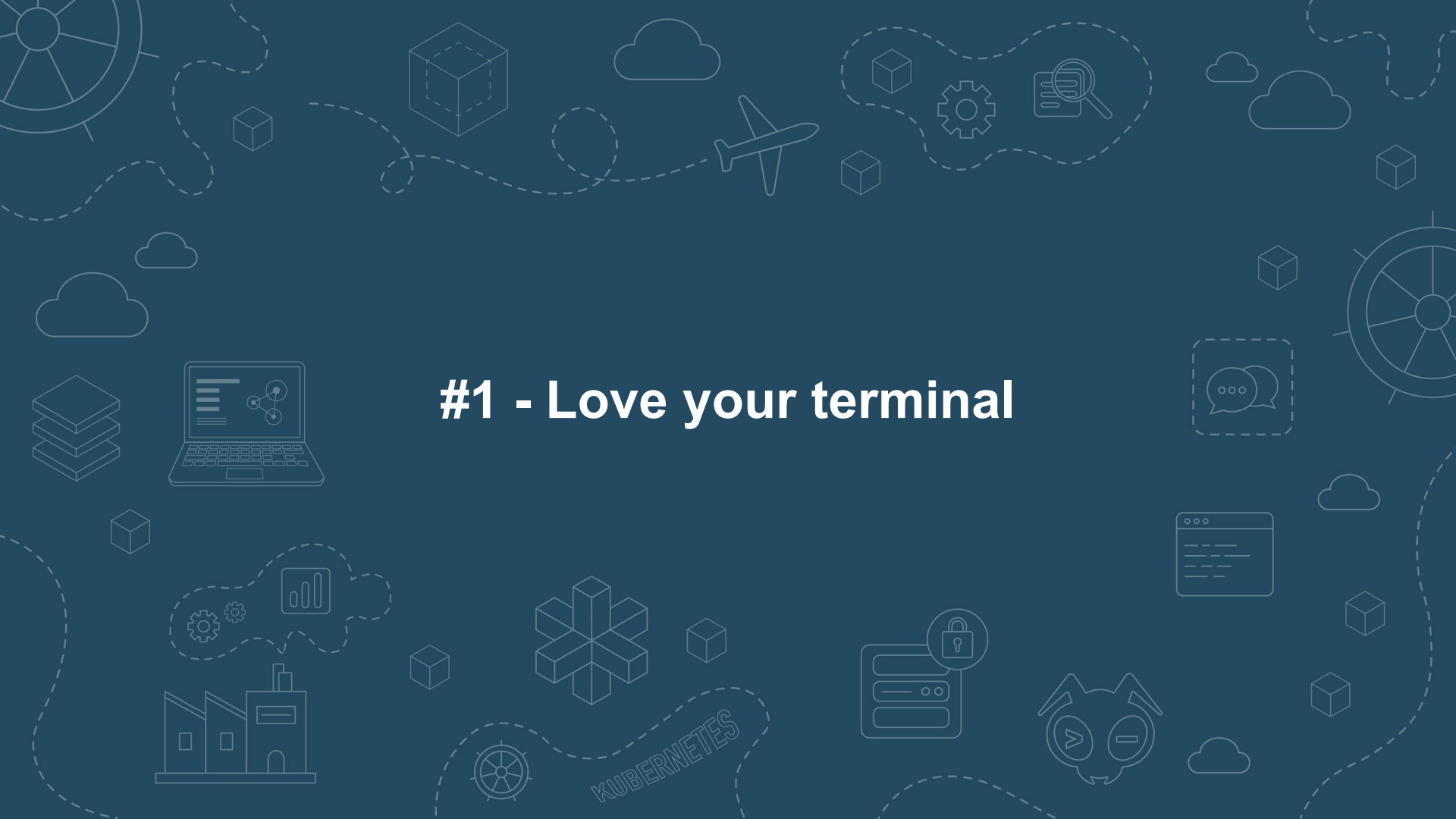
#8 → #10

Good have some programming knowledge



#0 - Pay someone else to deal with it

#1 - Love your terminal



#1 - Love your terminal

- ★ Bash? ZSH? Fish? - Doesn't matter as long as you're comfortable with it.
- ★ “rc” files - `.bashrc`, `.zshrc`
These set runtime configuration for each terminal window you open.
- ★ `alias` - easily create your own terminal commands
- ★ Look for “dotfiles” on GitHub - e.g. <https://github.com/averagemarcus/dotfiles>

#2 - Learn to love `kubectl`



#2 - Learn to love `kubectl`

★ The official documentation offers a single page view of all built in commands: kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

★ Add `alias k='kubectl'` to your `.bashrc` / `.zshrc` / `.whateverrc`

```
k get pods -A
```

★ `kubectl explain` is your friend! Find out what any property of any Kubernetes resource is for.

```
k explain pods.spec.containers
```

```
KIND:      Pod  
VERSION:  v1
```

```
RESOURCE: containers <[]Object>
```

```
DESCRIPTION:
```


#3 - Multiple kubeconfigs



#3 - Multiple kubeconfigs

- ★ Quick switch between different Kubernetes contexts (clusters) and between different namespaces.
- ★ `kubectx` and `kubens` - <https://github.com/ahmetb/kubectx>
- ★ `kubewitch` - <https://github.com/danielfoehrKn/kubewitch>
- ★ `kubie` - <https://github.com/sbstp/kubie>

#4 - k9s



#4 - k9s

```
1 k9s | github
Context: admin@talos-cluster
Cluster: talos-cluster
User: admin@talos-cluster
K9s Rev: v0.25.7 ↗ v0.25.18
K8s Rev: v1.23.1
CPU: 11%
MEM: 47%

Pod(s) (kube-system) [46]
NAME: PF READY RESTARTS STATUS CPU MEM %CPU/R %CPU/L %MEM/R %MEM/L IP NODE
coredns-6ff7786fb-8kn9r 1/1 3 Running 5 21 5 n/a 21 12 10.244.2.35 talos-19
coredns-6ff7786fb-mcd4 1/1 24 Running 9 19 9 n/a 28 11 10.244.1.145 talos-19
csi-smb-controller-57f6b9949d-lwz9b 3/3 76 Running 2 23 6 0 38 5 10.244.0.166 talos-19
csi-smb-controller-57f6b9949d-tmcd2 3/3 60 Running 5 30 16 1 51 7 10.244.1.147 talos-19
csi-smb-node-2j6bv 3/3 114 Running 2 31 6 0 52 7 192.168.1.197 talos-19
csi-smb-node-6w5kr 3/3 28 Running 1 37 3 0 61 9 192.168.1.19 talos-19
csi-smb-node-8ffah 3/3 68 Running 1 30 3 0 51 7 192.168.1.138 talos-19
csi-smb-node-qjfk 3/3 102 Running 1 27 3 0 45 6 192.168.1.126 talos-19
csi-smb-node-s2b5b 3/3 21 Running 1 18 3 0 30 4 192.168.1.207 talos-19
csi-smb-node-vkbgq 3/3 96 Running 1 18 3 0 30 4 192.168.1.177 talos-19
csi-smb-node-xn878 3/3 126 Running 1 17 3 0 29 4 192.168.1.191 talos-19
kube-apiserver-talos-192-168-1-177 1/1 25 Running 254 571 127 n/a 111 n/a 192.168.1.177 talos-19
kube-apiserver-talos-192-168-1-191 1/1 25 Running 175 494 87 n/a 96 n/a 192.168.1.191 talos-19
kube-apiserver-talos-192-168-1-207 1/1 4 Running 136 630 68 n/a 123 n/a 192.168.1.207 talos-19

[namespace] [pod]
```

github.com/derailed/k9s

#5 - kubectl plugins



#5 - kubectl plugins

- ★ Krew - package manager for kubectl plugins - <https://github.com/kubernetes-sigs/krew>
- ★ Any command in your `$PATH` that is prefixed with `kubectl-` becomes a kubectl plugin
- ★ Install plugins with `kubectl krew install <PLUGIN NAME>`
- ★ Some of my fave plugins:
 - stern - Multi-pod/container log tailing
 - tree - Show hierarchy of resources based on ownerReferences
 - outdated - Find containers with outdated images
 - gs - Giant Swarm's plugin for working with our managed clusters

Summary

My 10 tips for working with Kubernetes



~~#1 → #5~~

~~Anyone can start using these today~~

#6 → #7

Good to know a little old-skool ops first

#8 → #10

Good have some programming knowledge

#6 - kshell / kubectl debug



#6 - kshell / kubectl debug

Launch a temporary pod running a bash shell for cluster debugging

```
alias kshell='kubectl run \  
  -it \  
    Need more tools? Replace this with ubuntu ↘ \  
  --image bash \  
  --restart Never \  
  --rm \  
  shell'
```

#6 - kshell / kubectl debug

Launch a temporary pod running a bash shell for cluster debugging

```
# kshell
```

```
If you don't see a command prompt, try pressing enter.
```

```
bash-5.1# nslookup google.com
```

```
Server:          1.1.1.1
```

```
Address:         1.1.1.1:53
```

```
Non-authoritative answer:
```

```
Name:   google.com
```

```
Address: 142.250.187.206
```

#6 - kshell / kubectl debug

Debugging a running pod - `kubectl exec`

```
# kubectl exec my-broken-pod -it -- sh
error: Internal error occurred: error executing command in
container: failed to exec in container: failed to start exec.....
```

Debugging a running pod - `kubectl debug` [←] *Requires Kubernetes 1.23*

```
# kubectl debug -it --image bash my-broken-pod
Defaulting debug container name to debugger-gprmk.
If you don't see a command prompt, try pressing enter.
bash-5.1#
```

#6 - kshell / kubectl debug

Example - investigate a CrashLooping pod

```
# kubectl run debug-demo --image=bash -- exit 1
```

```
# kubectl get pods debug-demo
```

NAME	READY	STATUS	RESTARTS	AGE
debug-demo	0/1	CrashLoopBackOff	2 (20s ago)	44s

This will prevent us from 'kubectl exec' into the pod ↗

```
# kubectl debug -it --image bash debug-demo
```





Defaulting debug container name to debugger-5mkjj.

If you don't see a command prompt, try pressing enter.

```
bash-5.1#
```

#6 - kshell / kubectl debug

When to use what:

	kshell	kubectl exec	kubectl debug
Multiple workloads experiencing network issues			
Workload not running as expected but not CrashLooping and isn't a stripped down image (e.g. not Scratch / Distroless)			
Workload not running as expected but not CrashLooping and has an image based on Scratch / Distroless or similar			
Workload is CrashLooping			

#7 - kube-ssh



#7 - kube-ssh

★ <https://github.com/AverageMarcus/kube-ssh>

★ Give ssh-like access to a node host, great for instances where nodes are provisioned without SSH or direct access

```
sh -c "$(curl -sSL https://raw.githubusercontent.com/AverageMarcus/kube-ssh/master/ssh.sh)"  
[0] - ip-10-18-21-146.eu-west-1.compute.internal  
[1] - ip-10-18-21-234.eu-west-1.compute.internal  
[2] - ip-10-18-21-96.eu-west-1.compute.internal  
Which node would you like to connect to? 1
```

Please verify any script before you execute it! ↗

If you don't see a command prompt, try pressing enter.
[root@ip-10-18-21-234 ~]#

Why? - I prefer to use ephemeral instances with minimal needed to run Kubernetes, no sshd, no port 22 open etc. but there are times when you just need to check what's actually going on with the underlying host machine.

#7 - kube-ssh

How it works

ssh.sh

```
kubectl run kube-ssh --restart=Never -it --rm --image overridden
--overrides '
{
  "spec": {
    "hostPID": true,
    "hostNetwork": true,
    "${NODE_SELECTOR}"
    "tolerations": [{ "operator": "Exists" }],
    "containers": [
      {
        "name": "kube-ssh",
        "image": "averagemarcus/kube-ssh:latest",
        "stdin": true,
        "tty": true,
        "securityContext": { "privileged": true }
      }
    ]
  }
}' --attach "$@"
```

↪ Ensure we can run on any node

↪ Container image containing 'nsenter'

↪ Allows us to switch to a host PID

Dockerfile

```
FROM debian:buster as builder

WORKDIR /tmp
RUN apt-get update && \
  apt-get install -yq \
  make gcc gettext autopoint \
  bison libtool automake pkg-config

ADD https://github.com/karelzak/util-linux/archive/v2.34.tar.gz .
RUN tar -xf v2.34.tar.gz && \
  mv util-linux-2.34 util-linux \
  cd util-linux && \
  ./autogen.sh && \
  ./configure && \
  make LDFLAGS="--static" nsenter

FROM scratch
COPY --from=builder /tmp/util-linux/nsenter /
ENTRYPOINT [
  "/nsenter", "--all", "--target=1", "--", "su", "-"
]
```


#7 - kube-ssh

- ★ Some caveats - underlying host needs a shell ↩ This won't work on Talos, for example
- ★ You require enough permissions to launch pods with privileged securityContext - RBAC, PSPs and Admission Controllers could all potentially block this. (This could also be considered a benefit to this approach over traditional SSH)
- ★ Not a real SSH session
- ★ `nsenter` - “The nsenter command executes program in the namespace(s) that are specified in the command-line options.” ([Man page](#))

Summary

My 10 tips for working with Kubernetes



~~#1 → #5~~

~~Anyone can start using these today~~



~~#6 → #7~~

~~Good to know a little old skool ops first~~

#8 → #10

Good have some programming knowledge

#8 - Webhooks



KUBERNETES

#8 - Webhooks

OK, actually 3 but we're ignoring CRD
conversion webhooks ↘

★ Two types of webhooks:

ValidatingWebhook	Ability to block actions against the API server if fails to meet given criteria.
MutatingWebhook	Modify requests before passing them on to the API server.

- ★ Implement more advanced access control than is possible with RBAC. [[Restricting cluster-admin permissions](#)]
- ★ Add default labels to resources as they're created.
- ★ Enforce policies such as not using **latest** as an image tag or ensuring all workloads have resource requests/limits specified.
- ★ “Hotfix” for security issues (e.g. mutating all pods to include a **LOG4J_FORMAT_MSG_NO_LOOKUPS** env var to prevent Log4Shell exploit). [[Log4Shell Mitigation](#)]

#8 - Webhooks

- ★ Build your own operator to implement custom logic
- ★ [Kyverno](#) - Kubernetes native policy management. Create `Policy` and `ClusterPolicy` resources to define rules in YAML
- ★ [OPA Gatekeeper](#) - Policy management built on top of Open Policy Agent

Kyverno Policy

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: block-bulk-certconfigs-delete
  annotations:
    policies.kyverno.io/description: Block delete all bug in CLI
spec:
  rules:
    - name: block-bulk-certconfigs-delete
      match:
        any:
          - resources:
              kinds: [CertConfig]
      preconditions:
        any:
          - key: ""
            operator: Equals
            value: ""
      validate:
        message: |
          Your current kubectl-gs version contains a critical bug
        deny:
          conditions:
            - key: ""
              operator: In
              value: [DELETE]
```

Taken from my ["Restricting cluster-admin permissions"](#) blog
post ↗

#8 - Webhooks

Notes:

This is one of the main causes we see of clusters being down ↴

- ★ Where possible always avoid applying webhooks to resources in `kube-system`. This can cause a deadlock if those pods try to come up before the webhook service is available.
- ★ Be aware of the `failurePolicy` property - it defaults to “fail” which can cause troubles if your service handling the webhook goes down.
- ★ The `reinvocationPolicy` property can be set if changes made by a `MutatingWebhook` may need to go through other defined webhooks again.
- ★ Ordering - first `MutatingWebhooks` then `ValidatingWebhooks`. No guaranteed control of order within these two phases.

#9 - Kubernetes API



#9 - Kubernetes API

All Kubernetes operations are done via the API - kubectl uses it, in-cluster controllers use it, the scheduler uses it and ***you can use it too!*** ✨

Currently using OpenAPI V2 (OpenAPI V3 available as an alpha feature in v1.23)

The API can be extended either by Custom Resource Definitions (CRDs) or by implementing an Aggregation Layer (such as what metrics-server implements).

#9 - Kubernetes API

You can easily try out the API using `kubectl` with the `--raw` argument.

This is the equivalent to 'kubectl get pods -n default' ↴

```
# kubectl get --raw /api/v1/namespaces/default/pods
{"kind":"PodList","apiVersion":"v1","metadata":{"selfLink":...
```

If no host is provided `kubectl` will use the API of the current context.

HTTP Method	Kubectl command
GET	<code>kubectl get --raw</code>
POST	<code>kubectl create --raw</code>
DELETE	<code>kubectl delete --raw</code>
PUT	<code>kubectl replace --raw</code>

#9 - Kubernetes API

Not sure what APIs are available?

```
# kubectl api-resources
```

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
bindings		v1	true	Binding
componentstatuses	cs	v1	false	ComponentStatus
configmaps	cm	v1	true	ConfigMap
endpoints	ep	v1	true	Endpoints
deployments	deploy	apps/v1	true	Deployment

API endpoint format:

From the "Name" column above

```
/ {API_VERSION} / namespace / {NAMESPACE_NAME} / {RESOURCE_KIND} / {NAME}
```

Only if namespaced column is 'true' ↗

Optional name of the specific resource



#9 - Kubernetes API

Not sure what APIs are available?

```
# kubectl api-resources
```

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
bindings		v1	true	Binding
componentstatuses	cs	v1	false	ComponentStatus
configmaps	cm	v1	true	ConfigMap
endpoints	ep	v1	true	Endpoints
deployments	deploy	apps/v1	true	Deployment

This is the "core" API ↗

If `APIVERSION` is just v1 the endpoint starts with `/api/v1/`

E.g. `/api/v1/componentstatuses`

#9 - Kubernetes API

Not sure what APIs are available?

```
# kubectl api-resources
```

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
bindings		v1	true	Binding
componentstatuses	cs	v1	false	ComponentStatus
configmaps	cm	v1	true	ConfigMap
endpoints	ep	v1	true	Endpoints
deployments	deploy	apps/v1	true	Deployment

Otherwise, the endpoint starts with `/apis/{APIVERSION}/`

Note the extra 's' ↗

E.g. `/apis/apps/v1/`

#9 - Kubernetes API

Not sure what APIs are available?

```
# kubectl api-resources
```

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
bindings		v1	true	Binding
componentstatuses	cs	v1	false	ComponentStatus
configmaps	cm	v1	true	ConfigMap
endpoints	ep	v1	true	Endpoints
deployments	deploy	apps/v1	true	Deployment

The **NAMESPACED** column indicates if the resource is bound to a namespace.

If false: `/api/v1/componentstatuses` *← Name of the namespace to use*

If true: `/apis/apps/v1/namespaces/default/deployment`

Omit this to search all namespaces



#9 - Kubernetes API

Resources:

- [kubernetes/client-go](#) - the official Golang module for interacting with the Kubernetes API
- [Kubernetes Provider](#) for Terraform (actually uses the above Go module under the hood)
- [kubernetes-client/python](#) - the official Python library for interacting with the Kubernetes API

Where is this useful?

- ★ Building our own CLI / desktop tooling (e.g. k9s, Lens).
- ★ Cluster automation - resources managed by CI, CronJobs, etc.
- ★ Building our own operators to extend Kubernetes.

#10 - CRDs & Operators



#10 - CRDs & Operators

Extend Kubernetes' built-in API and functionality with your own Custom Resource Definitions (CRDs) and business logic (operators).

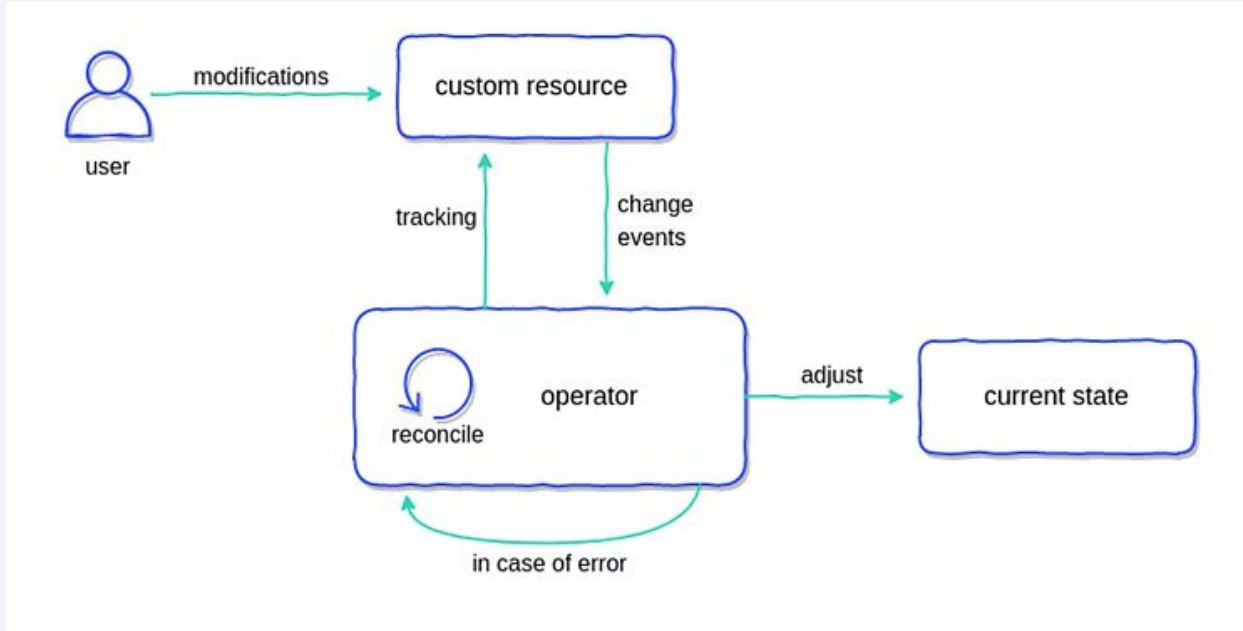
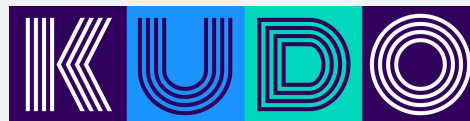


Image credits: Container Solutions

<https://blog.container-solutions.com/kubernetes-operators-explained>

#10 - CRDs & Operators

Frameworks



[Metacontroller](#)

References

- <https://kubernetes.io/docs/concepts/extend-kubernetes/operator/>
- <https://blog.container-solutions.com/kubernetes-operators-explained>
- <https://operatorhub.io/> - Directory of existing operators

Videos



Summary

My 10 tips for working with Kubernetes



~~#1 → #5~~

~~Anyone can start using these today~~



~~#6 → #7~~

~~Good to know a little old skool ops first~~



~~#8 → #10~~

~~Good have some programming knowledge~~

Recap

#1 - Love your terminal

#2 - Learn to love kubectl

#3 - Multiple kubeconfigs

#4 - k9s

#5 - Kubectl plugins

#6 - kshell / kubectl debug

#7 - kube-ssh

#8 - Webhooks

#9 - Kubernetes API

#10 - CRDs & Controllers

Thank You



Slides available at:

<https://go-get.link/devopsnotts22>