KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

KubeCon CloudNativeCon
Europe 2022

Kubectl Said What?!?

Christopher Hanson, RX-M, llc.

whoami

Christopher Hanson (Chris)
Senior Cloud Native Engineer at RX-M, llc.

Instructor – Kubernetes, Helm, Prometheus, Argo, Spinnaker, Docker, OpenShift, Concourse, Ansible, OpenStack (among others)

Consultant / **Advisor** – monolith to containers/K8s app migrations, K8s Cassandra DBaaS, monitoring/logging pipeline using the EFK stack

Speaker
KubeCon / CloudNativeCon Europe Virtual 2021:
Choose Wisely: Understanding Kubernetes Selectors
(CNCF YouTube: https://youtu.be/dLe0TZEGhxo)

KubeCon / CloudNativeCon Europe Virtual 2020:
Zero Downtime Deployments: Controlling Application Rollouts and Rollbacks
(CNCF YouTube: https://youtu.be/rh6EtXiNOj4)

San Francisco Meetup: Managing Resources in K8s apps
(RX-M YouTube channel: https://youtu.be/ijWZinxdHEA)

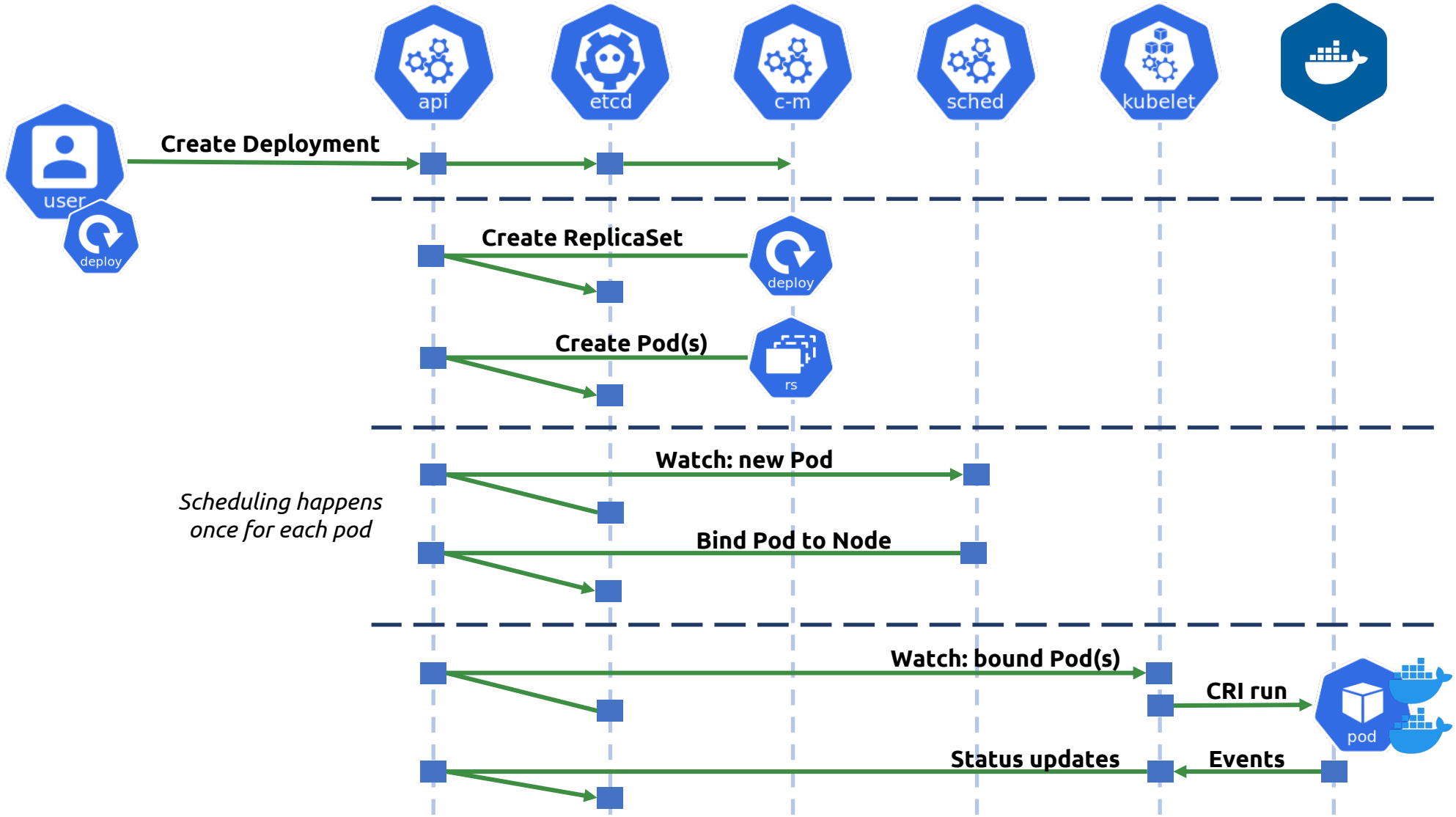
Exam Developer: Certified Kubernetes Security Specialist (CKS)
Certified Kubernetes Administrator (CKA-1700-0131-0100)
Certified Kubernetes Application Developer (CKAD-1900-0994-0100)

KubeCon | CloudNativeCon
Europe 2022

@CloudNativChris
chris@rx-m.com

rx-m cloud native training & consulting
rx-m.com / info@rx-m.com

Resource Creation Flow



Deciphering kubectl's Messages

Kubernetes resources have "conditions", "phases", "states", and "status"

- Some are summaries, others are detailed
- All can provide clues to the state of your resources

When first learning K8s, it can be difficult (even frustrating) to find the information you need, when you need it

- Where do I find it?
- How is it presented?
- Some output can be noisy—how do I avoid red herrings?

"I just dropped in to see what condition my condition was in"

– 

Information about resource states can be viewed using commands like:

- `kubectl get events | <object>`
- `kubectl get` with modifiers like `--output` (`-o`):
  - `kubectl get -o yaml|json`
  - `kubectl get -o jsonpath` under `status.conditions[*].message`
- `kubectl describe`

Let's examine what these commands present with some practical examples…

Controllers: Conditions

Controllers track conditions related to pods under their control

In *some cases*, can be seen with **`kubectl describe`**

- Conditions can be transient

Deployment

- `Available` – minimum replicas are up and running for at least `minReadySeconds`
- `Progressing` – new RS(es) created, scale up/down
- `ReplicaFailure` – added when one of its pods fails to be created/or deleted

ReplicaSet

- `ReplicaFailure` – added when one of its pods fails to be created/or deleted

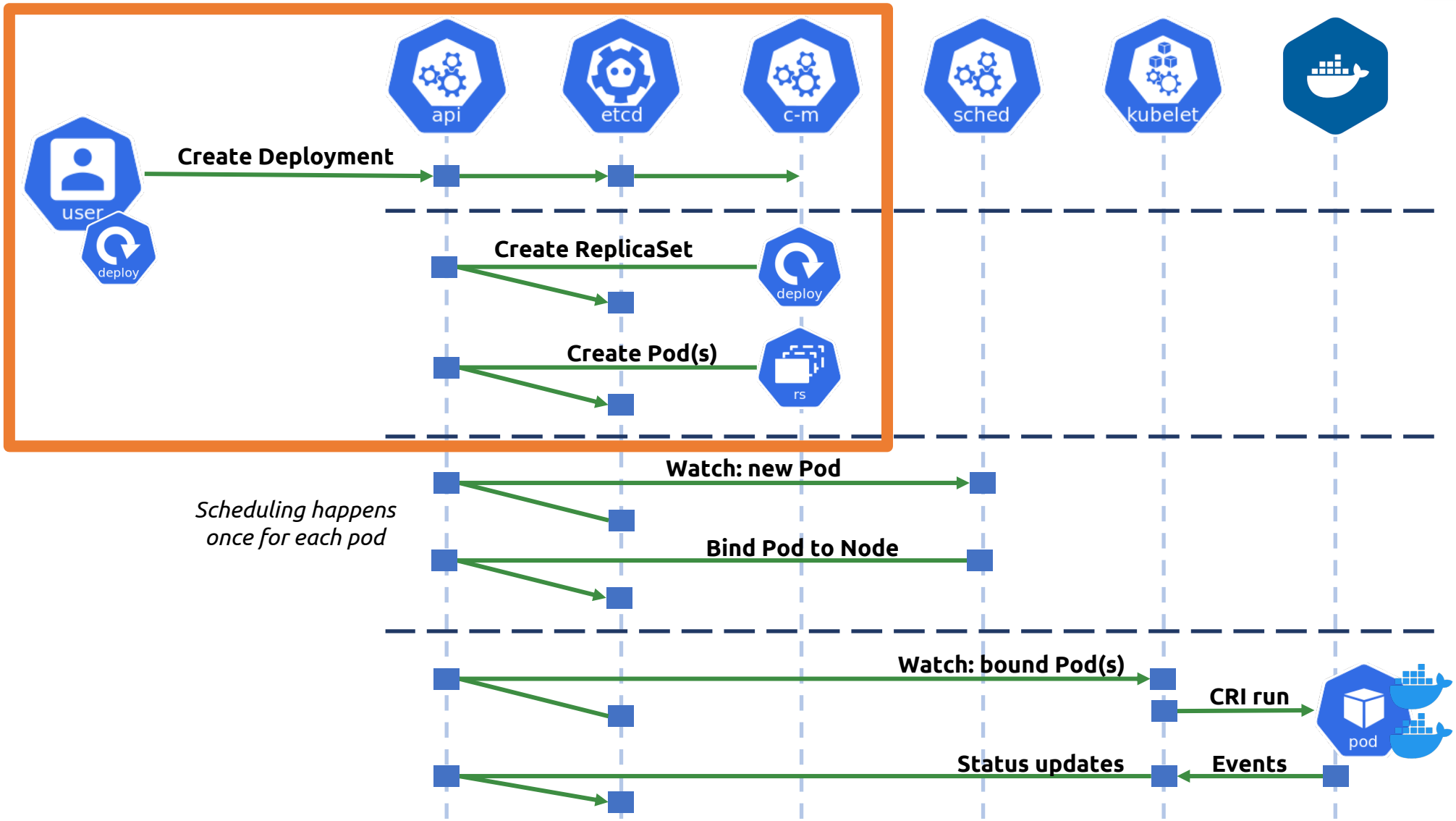
StatefulSet

- `statefulset.status.conditions` – is part of the object but is not presented by kubectl?

Job

- `conditions.type` – `Complete` or `Failed` based on the pod(s) exit code(s)
  - If still running, "`conditions`" will not be present

Controllers: Example



Failing to Create Pods

Symptoms:

Deployments / StatefulSets will have mismatches between available/ready replicas
- Will look identical to other issues – ***may be misleading!***

ReplicaSets will reflect accurate desired/current/ready replicas
- If no other problems exist, ready = current ≠ desired

Pods are missing – pods that can't be created will not have any pod object to query for more info

Errors will populate a related condition "`message`" field
(depending on controller)

Common Cause:

Quota – namespace where a controller is creating pods has a resource quota in effect, blocking pod(s) from being created

Pods & Containers

When the controller *can* create pod(s), the pod object exists and can be queried for more info

Pods: Phases

Pod Phases are **macro states** in the lifecycle of a Kubernetes pod

Seen when using **`kubectl get pod`** but also available from a pod's `status.phase`

`Pending` – pod accepted by the system
- >=1 of the containers has *not* been created
- Includes time before scheduling and time downloading images

`Running` – pod has been bound to a node (scheduled)
- All of a pod's containers have been created
- >=1 container is running, starting, or restarting

`Succeeded` – all container(s) in a pod have terminated in success (zero exit) and will not be restarted

`Failed` – >=1 container(s) terminated in failure (non-zero exit or terminated by the cluster)

`Unknown` – state of the pod cannot be obtained
- Most likely a communication error with the kubelet on the host where the pod is scheduled/running

Pods: Conditions

Array of conditions the pod has / has not passed in its lifecycle

Can be easily found/viewed with `kubectl describe`

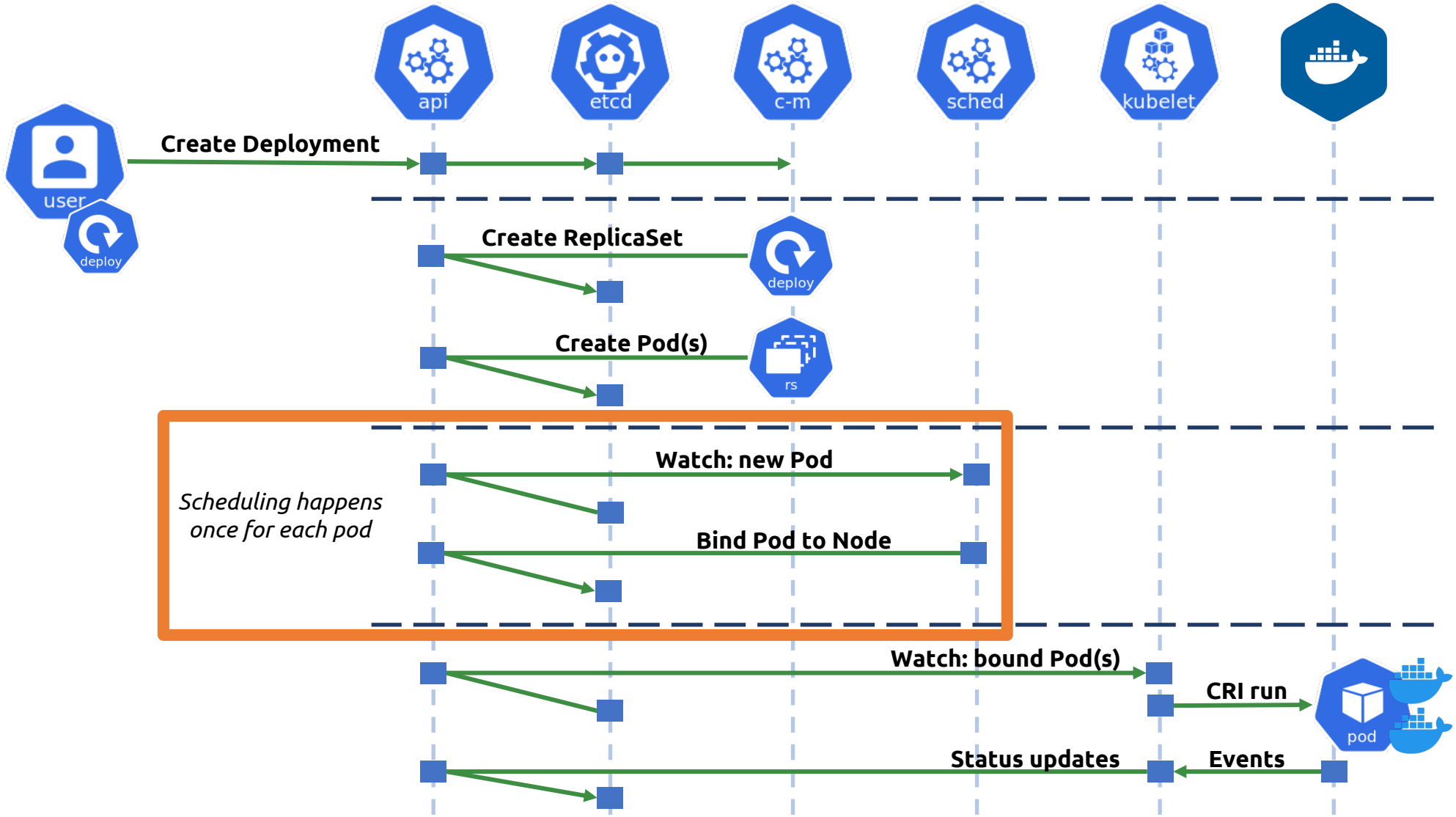
`Ready` – able / not able to serve client/peer requests, should be added to service/routing mesh

`Initialized` – all init containers completed successfully (zero exit) or are still running/restarting

`ContainersReady` – all containers in the Pod are / are not ready

`PodScheduled` – the pod has / has not been scheduled to a node

Pending Pods: Examples



Pending: Symptoms

Controllers will have mismatches between desired/current/ready replicas
- Will look identical to other issues – ***may be misleading!***

Pods will exist and `kubectl get pod` will return `Pending` (duh)

K8s will report the pod condition `PodScheduled` as "`false`"
- No other pod conditions will be present

Pod condition "`message`" field and related events will provide more details about the cause

Container states will be null – no containers are created unless a pod is scheduled to a node

Pending: Common Causes

Insufficient resources – container request(s) cannot be met (CPU, memory, disk)
- Ignore "limits" as these *are not* used by the scheduler

Selectors – pod specifies `nodeName`, node label selector (`nodeSelector`), or node/pod affinity/anti-affinity that doesn't match any nodes or pods

Volumes – a pod's PVC cannot be fulfilled by any PVs, missing ConfigMap(s) and/or Secret(s)
- ConfigMaps & Secrets have additional messaging (covered later)

Priority Class Preemption – a running pod may be evicted in favor of another pod of a higher priority class
- Controller will attempt to replace the evicted pod with a replacement which will sit in pending state because resources are scarce
- **Let's examine this special case before moving on…**

Running Pods, Containers: States

Each container in a pod has its state tracked

Includes containers, initContainers, and ephemeralContainers (if enabled)

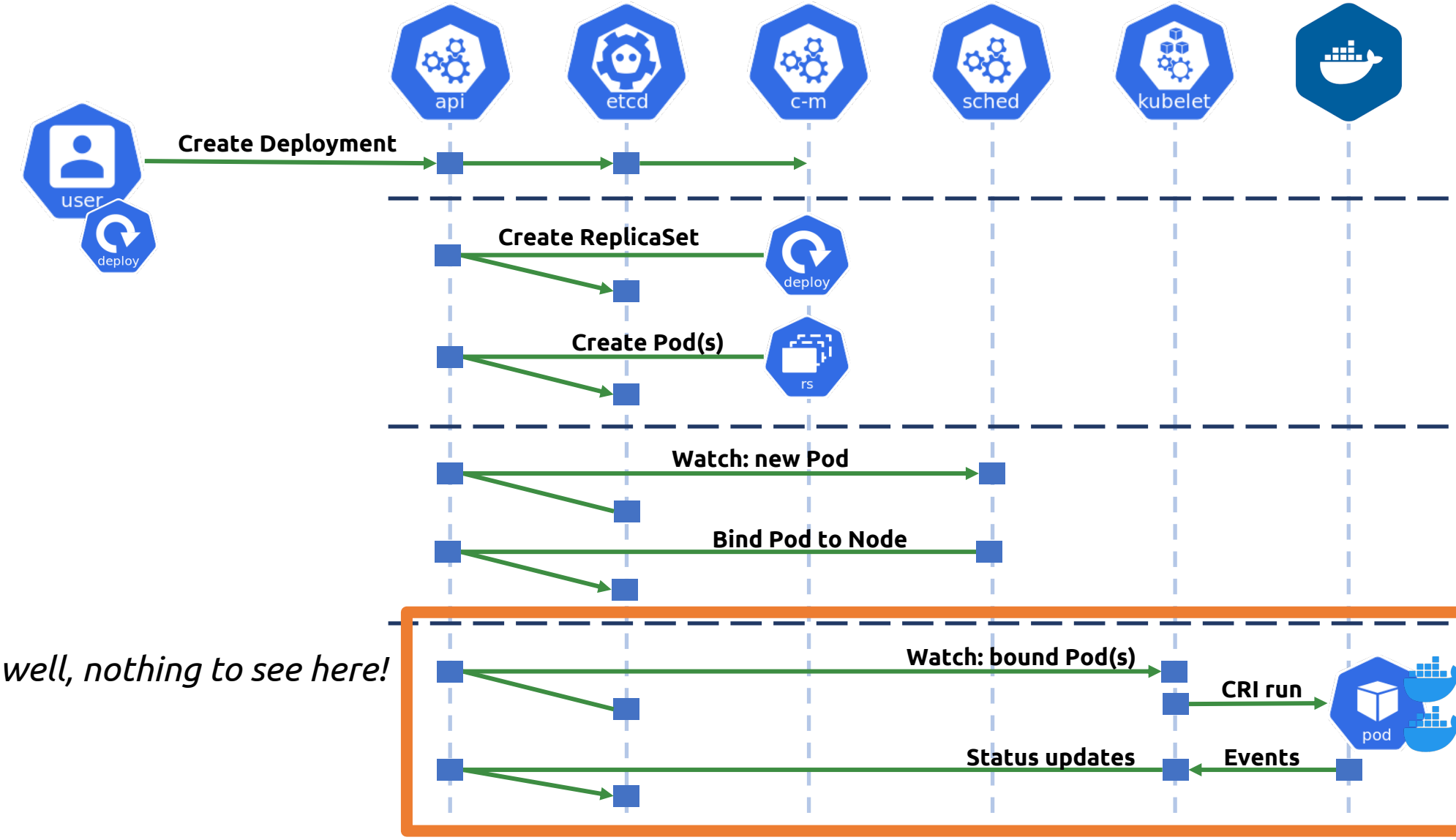
Can be viewed—under each container's name–with **`kubectl describe`**
- The **`containerStatuses`** list of **`kubectl get -o yaml`** is a better presentation/grouping of this info

`Waiting` – container is running operations to complete start up
- Includes image pulls and ConfigMap or Secret mounting
- Back-off time during CrashLoopBackOff

`Running` – container processes executing
- Any postStart hooks completed successfully

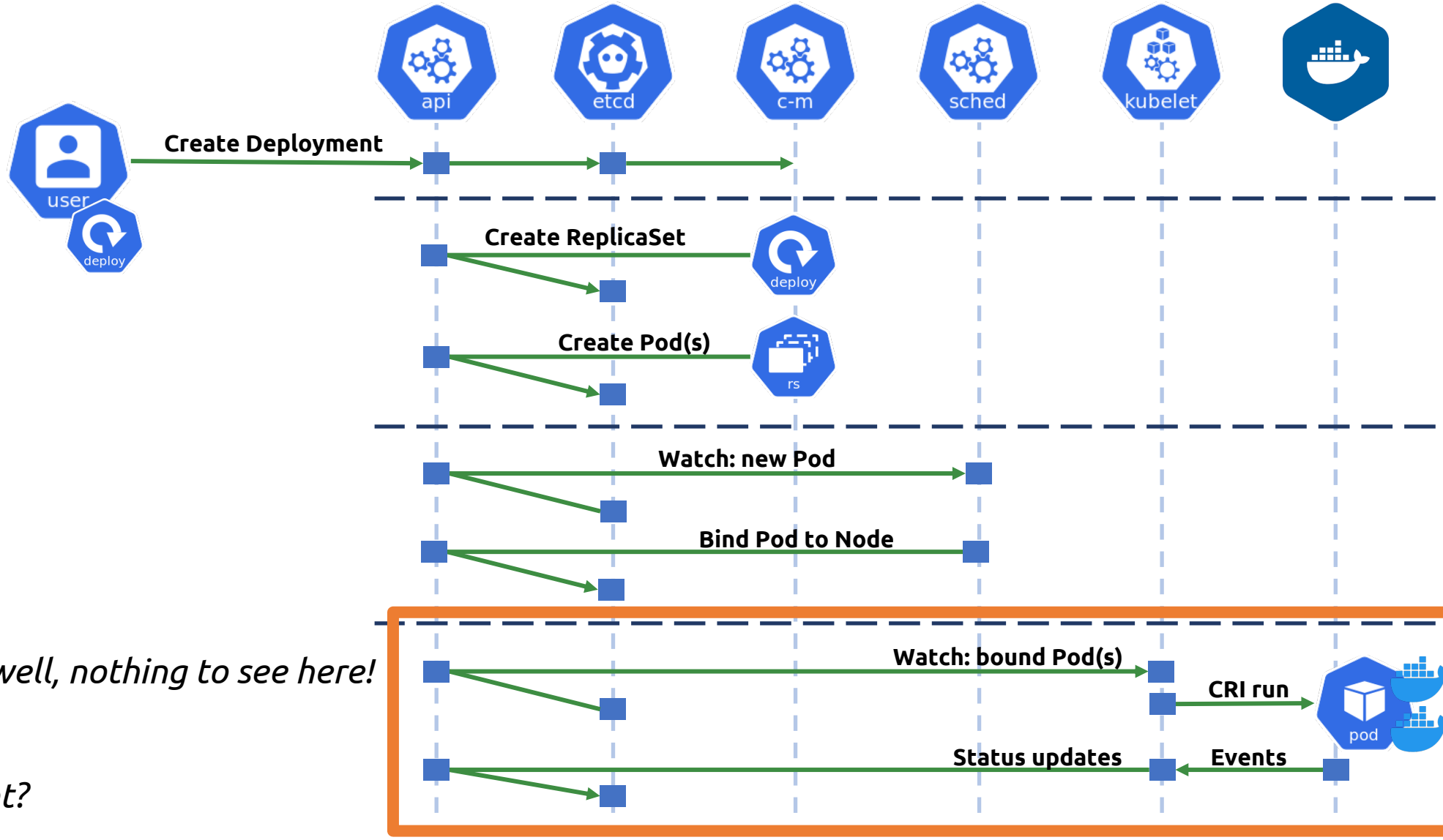
`Terminated` – container ran and failed or finished successfully
- Any preStop hooks executed before a container enters this state

Running: Examples



All is well, nothing to see here!

Running: Examples



All is well, nothing to see here!

…right?

Running (but not Ready): Symptoms

Controllers will have mismatches between desired/current/ready/available replicas
- Will look identical to other issues – ***may be misleading!***

`kubectl get pod` will return `Running` but `Ready` will not have equal values for current / desired

K8s will report the pod condition `PodScheduled` as "**`true`**"

K8s will report the pod conditions `Ready` and `ContainersReady` as "**`false`**"
- Condition messages will indicate which container(s) are problematic
  - Ex: `'containers with unready status: [container-names]'`

Container states will vary based on cause

Related: Services will have fewer endpoints than pods that are running

Running (but not Ready): Common Causes

Probes – one or more of the probes has failed
- Container state will be "`running`" but "`ready`" will be false

Errors / Crashing / Restarting – because >=1 container is running, starting, or restarting the pod remains in the "Running" phase
- When happening frequently, `Running` is replaced by `CrashLoopBackoff`
  - Container state will be "`waiting`" and "`ready`" will be false

Running and Ready (but not available)

KubeCon CloudNativeCon Europe 2022

Symptoms:

Controllers will look normal—at first!

- After the kubelet heartbeat timestamp decays (~40 seconds), `Ready` and `Available` will mismatch with `Desired`
  - Will look identical to other issues – ***may be misleading***

`kubectl get pod` returns **`Running`** and **`Ready`** until the pod is terminated—***will be misleading***

K8s will report the pod conditions **`PodScheduled`** and **`ContainersReady`** as "**`true`**"

K8s will report the pod condition **`Ready`** as "**`false`**" only *after* the node is considered not ready

Common Cause:

Node failure – the `Running` phase will not change but the node controller will indicate that the pod is not ready

- Status updates no longer being delivered by the kubelet (which is down)

Other Examples

When a pod's phase does not reflect:

- `Pending`
- `Running`
- `Succeeded`
- `Failed`
- `Unknown`

Pods with other Statuses

KubeCon CloudNativeCon Europe 2022

Even though a pod's `status.phase` will always be one of:

- `Pending`
- `Running`
- `Succeeded`
- `Failed`
- `Unknown`

`kubectl get pod` will show the value of `.status.containerStatuses[*].state.*.reason`

`Pending`:

- `ContainerCreating`
- `CreateContainerConfigError`
- `ErrImagePull` / `ImagePullBackOff`

`Running`:

- `Error` / `CrashLoopBackOff`

To help with debugging commonly occurring issues

Now you know what kubectl said!

Thank You!

KubeCon | CloudNativeCon Europe 2022

Cloud Native Short Takes: ~5 minute tips on a myriad of subjects: K8s, containers, Golang, etc. on the RX-M YouTube channel

Certification Training: KCNA / CKAD / CKA / CKS – self-study & virtual/in-person instructor-led training by a certified instructor

RX-M Cloud Native Expertise:
Training – Consulting / Advisory – Staffing

- Large breadth of cloud native / open source / digital transformation training curriculum
- Custom consultative training programs aligned to job roles and business goals

Contact us at: rx-m.com / info@rx-m.com

rx-m cloud native training & consulting

kubernetes TRAINING PARTNER

kubernetes CERTIFIED SERVICE PROVIDER

THE LINUX FOUNDATION AUTHORIZED TRAINING PARTNER

CLOUD NATIVE COMPUTING FOUNDATION SILVER MEMBER