Google

Kubernetes Design Principles

May 3, 2018

Saad Ali
Senior Software Engineer, Google
Co-Lead SIG-Storage
github.com/saad-ali
twitter.com/the_saad_ali

CAMPUS TEL AVIV

Agenda

- Illustrate principles of Kubernetes design, by showing how Kubernetes works.
- What's in it for me?
  - A deeper understanding of Kubernetes

What is Kubernetes?

- Containerization was the key
  - Consistent, repeatable, reliable deployments on a wide variety of systems.
- Who will manage it?
  - You? Scripts? A system you write?
- Kubernetes is a system for monitoring & deploying containerized workloads to nodes in a cluster.

Creating a Pod

- Principle: Kubernetes APIs are declarative rather then imperative.
- Create an API object (using CLI or REST API) to represent what you want to do.
- All the components in the system will work to drive towards that state, until the object is deleted.
- Example: Declare container “mycontainer” should be running.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image:
internal.mycorp.com:5000/mycontainer:1.7.9
```

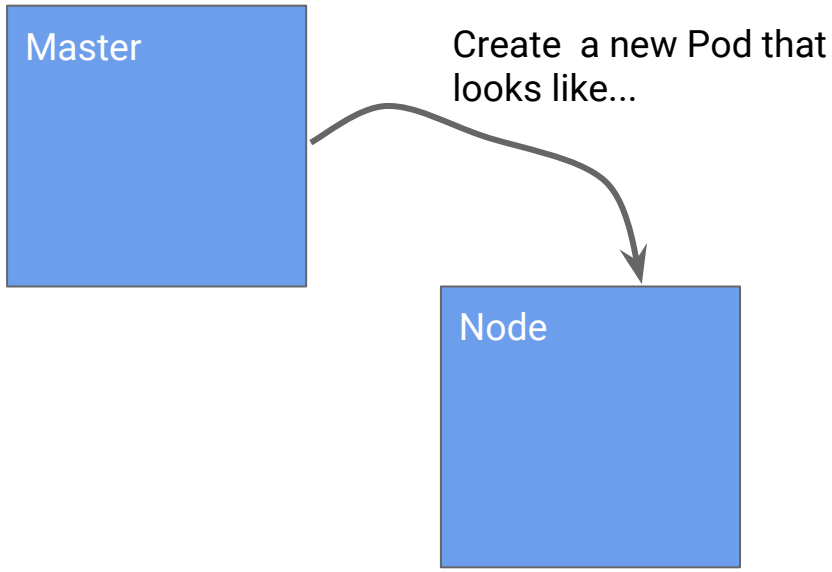
Why declarative over imperative?

More robust system that can easily recover from failure of components.

- No single point of failure.
- Components level triggered instead of edge triggered -- no “missing events” issues.

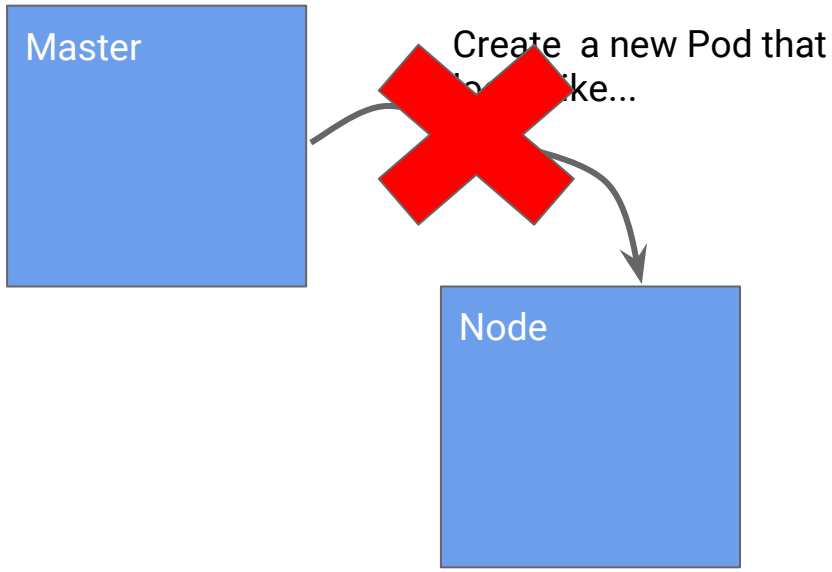
Creating a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.



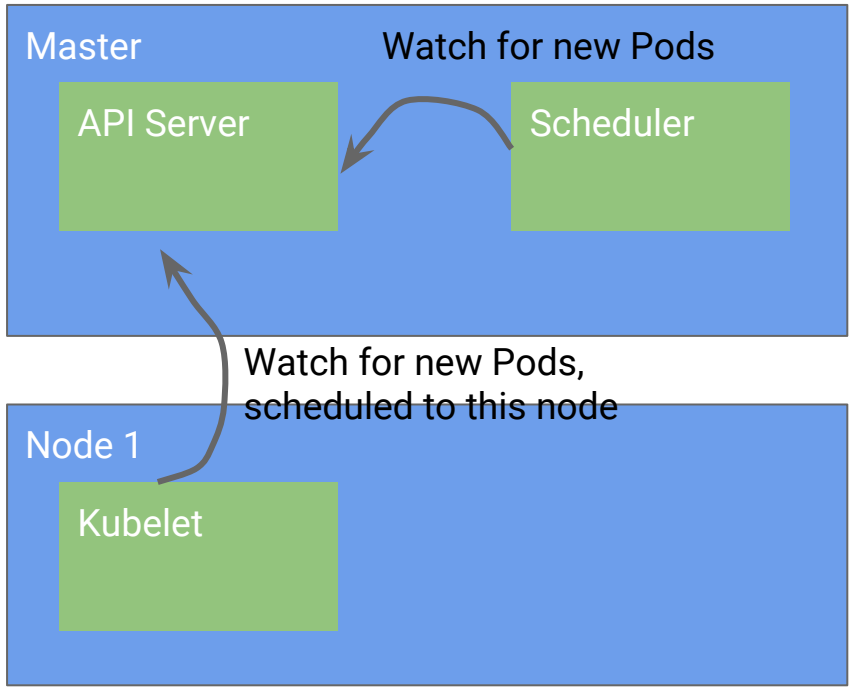
Creating a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.



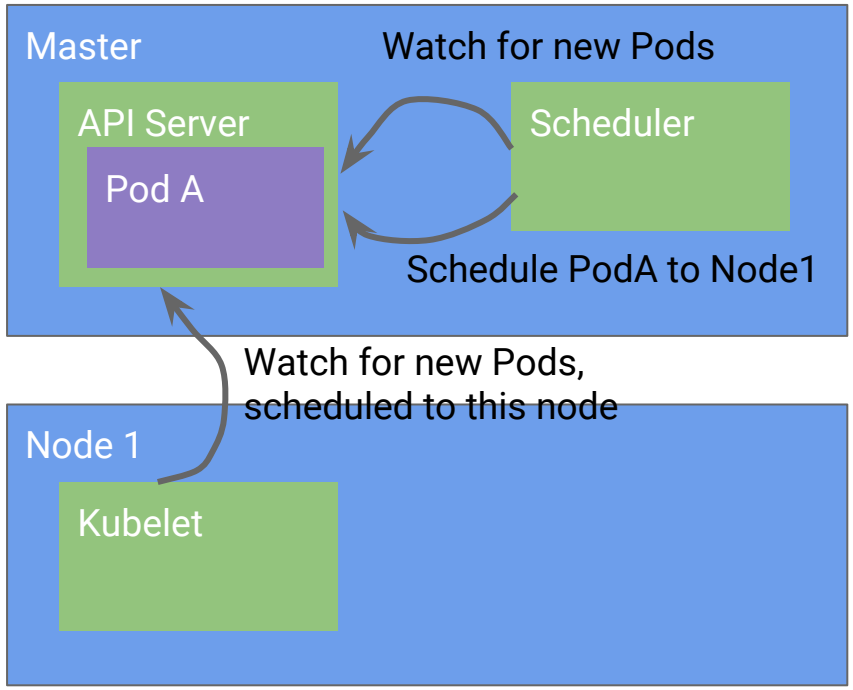
Creating a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.



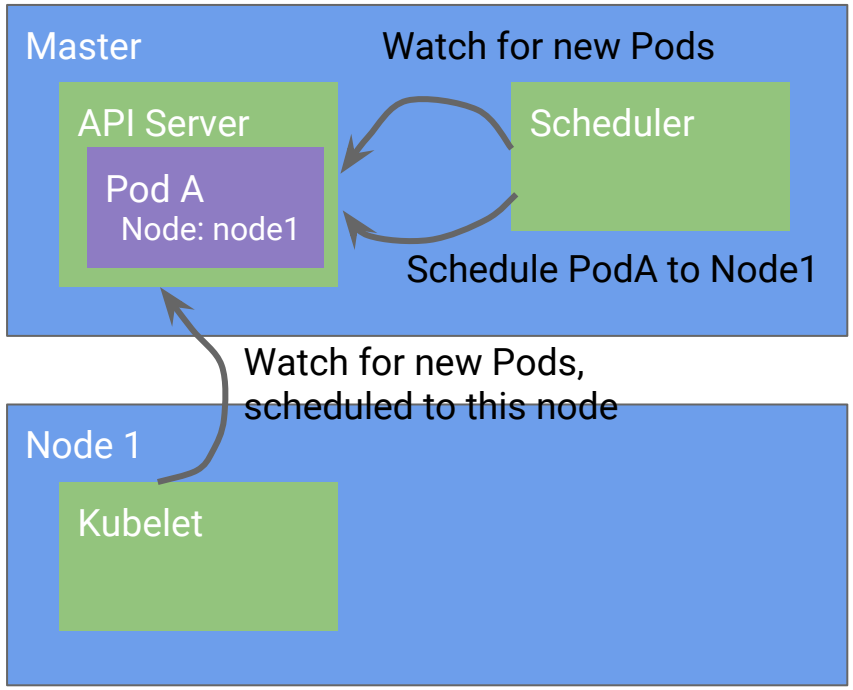
Creating a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.



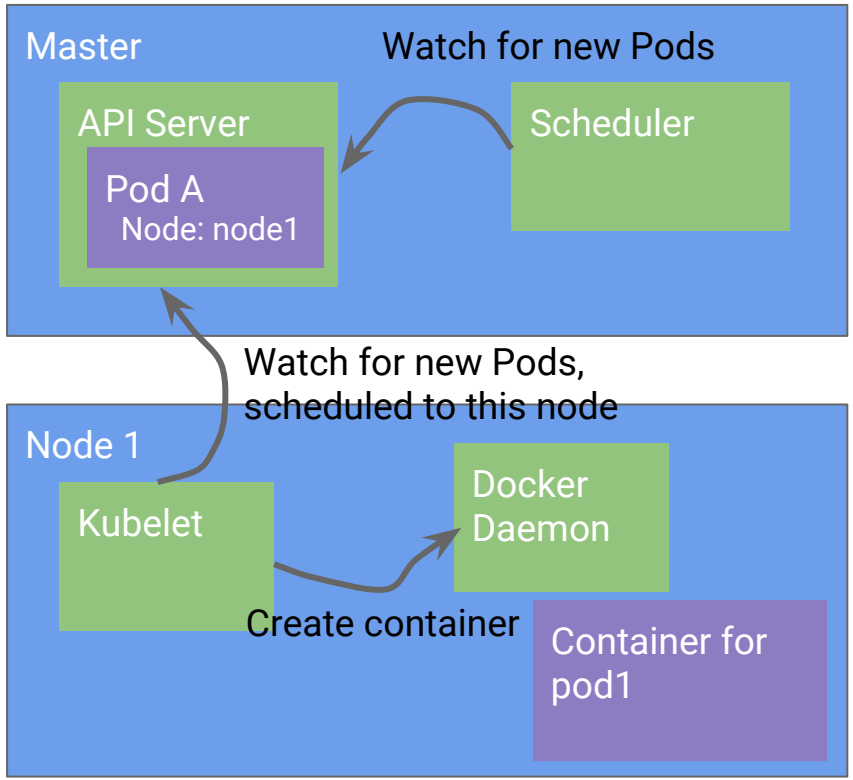
Creating a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.



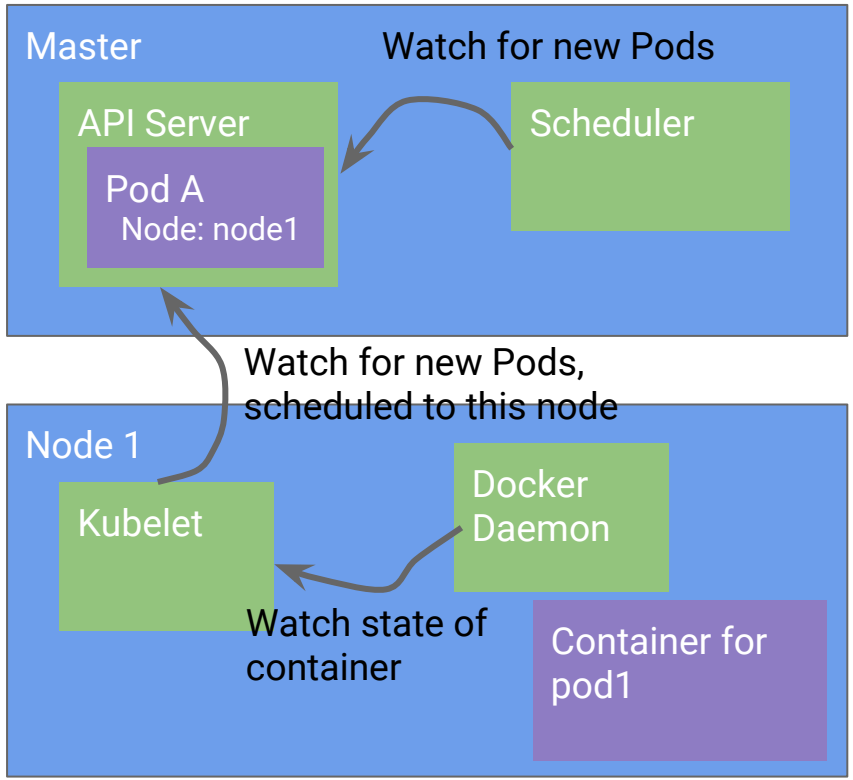
Creating a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.



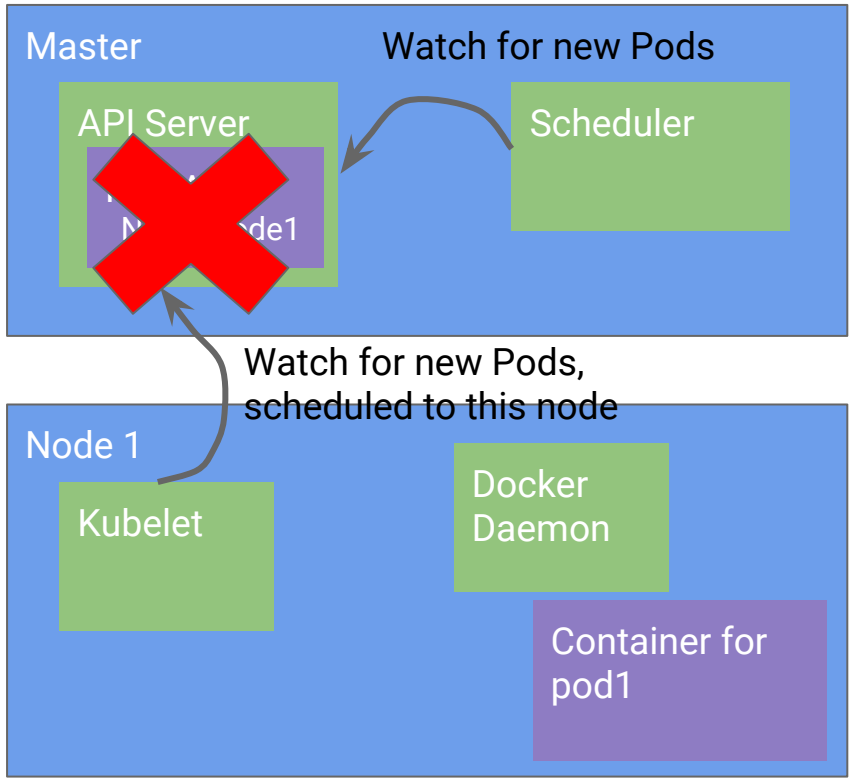
Creating a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.



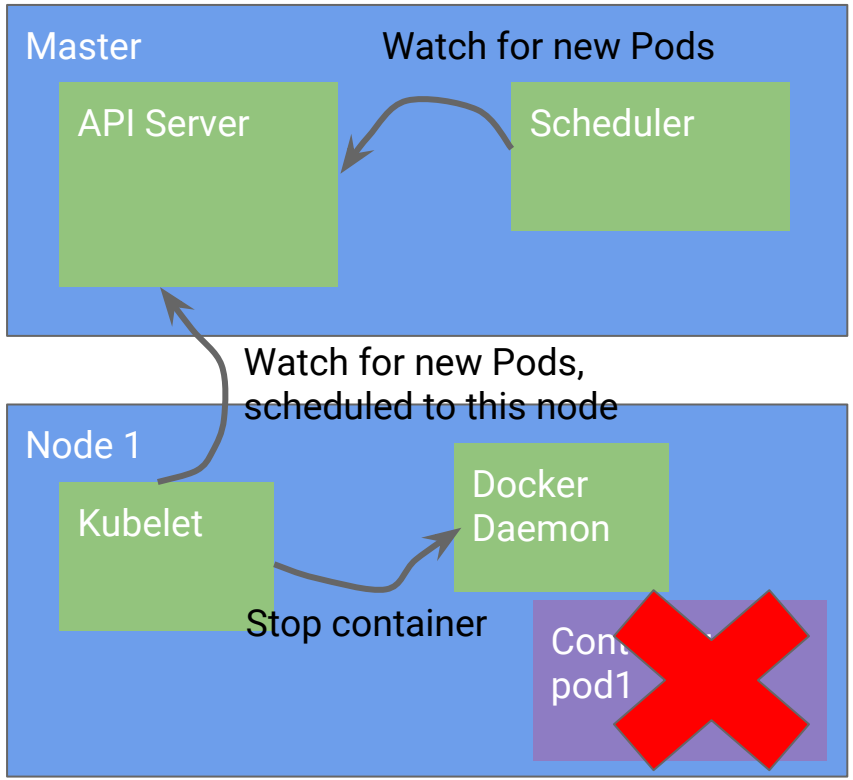
Stopping a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.
- To terminate a pod, you delete the pod object.
- Principle: Kubernetes APIs are declarative rather then imperative.



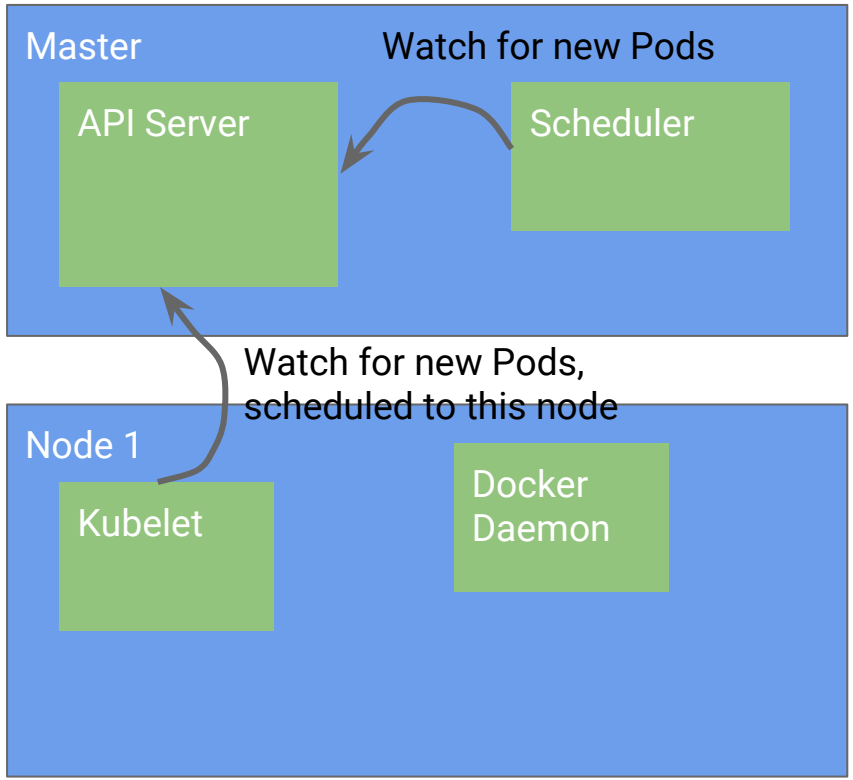
Stopping a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.
- To terminate a pod, you delete the pod object.
- Principle: Kubernetes APIs are declarative rather then imperative.



Stopping a Pod

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Every component watches the Kubernetes API and works to drive towards desired state.
- To terminate a pod, you delete the pod object.
- Principle: Kubernetes APIs are declarative rather then imperative.



Why no hidden internal APIs?

Makes Kubernetes composable and extensible.

- Default component not working for you?
  - Turn it off and replace it with your own.
- Additional functionality not yet available?
  - Write your own and to add it.

Kubernetes Volume Plugins

Kubernetes has many volume plugins

Remote Storage

- GCE Persistent Disk
- AWS Elastic Block Store
- Azure File Storage
- Azure Data Disk
- Dell EMC ScaleIO
- iSCSI
- Flocker
- NFS
- vSphere
- GlusterFS
- Ceph File and RBD
- Cinder
- Quobyte Volume
- FibreChannel
- VMware Photon PD

Ephemeral Storage

- EmptyDir
- Expose Kubernetes API
  - Secret
  - ConfigMap
  - DownwardAPI

Local Persistent Volume (Beta)

Out-of-Tree

- Flex (exec a binary)
- CSI (Beta)

Other

- Host path

Ephemeral storage

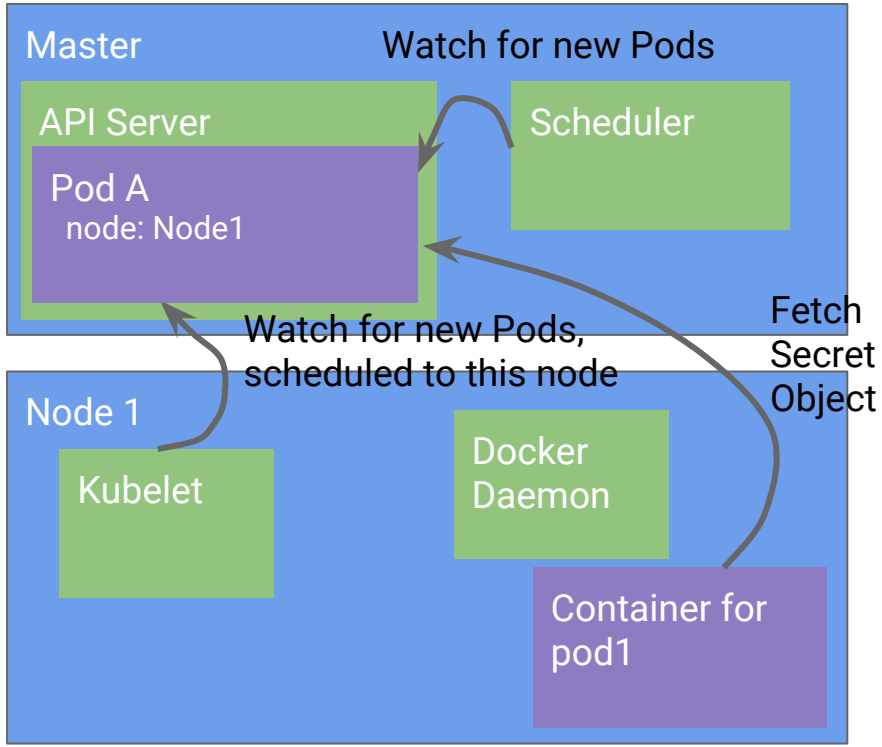
- Volume whose lifecycle is tied to the lifecycle of pod.
  - Temp empty scratch file space from host machine, when pod starts.
  - Deleted when pod is terminated.
- Enables sharing state between containers in a pod.
- Plugin: EmptyDir

Kube API Data

- Kubernetes API has lots of data that is interesting to workloads
  - Secrets - Sensitive info stored in KubeAPI
    - e.g. passwords, certificates, etc.
  - ConfigMap - Configuration info stored in KubeAPI
    - e.g. application startup parameters, etc.
  - DownwardAPI - Pod information in KubeAPI
    - e.g. name/namespace/uid of my current pod.

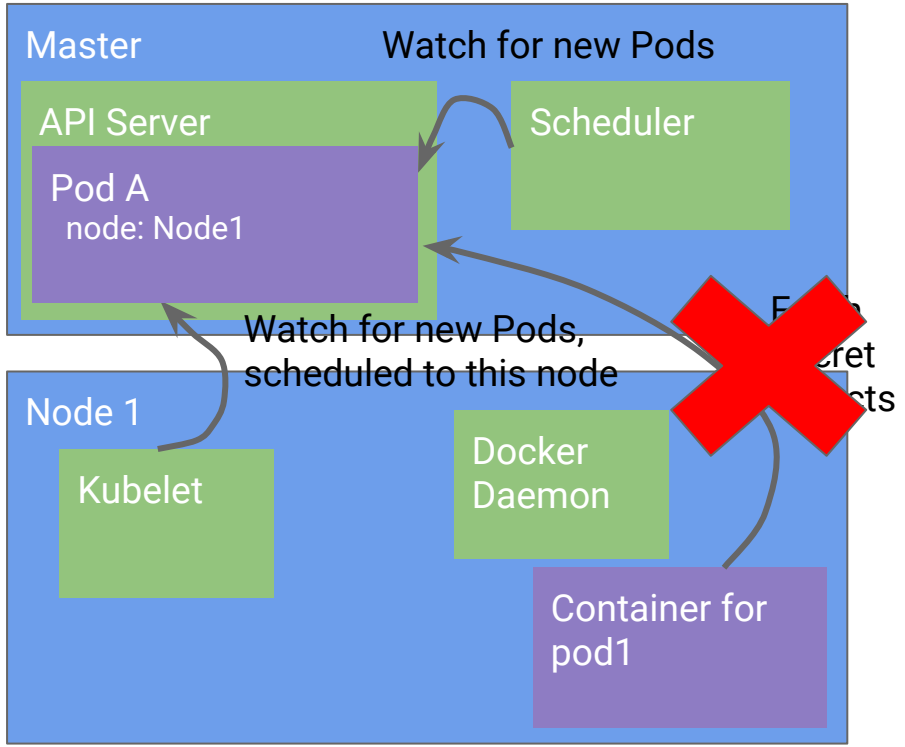
Fetching Kube API Data

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Could modify application to communicate directly with API Server.



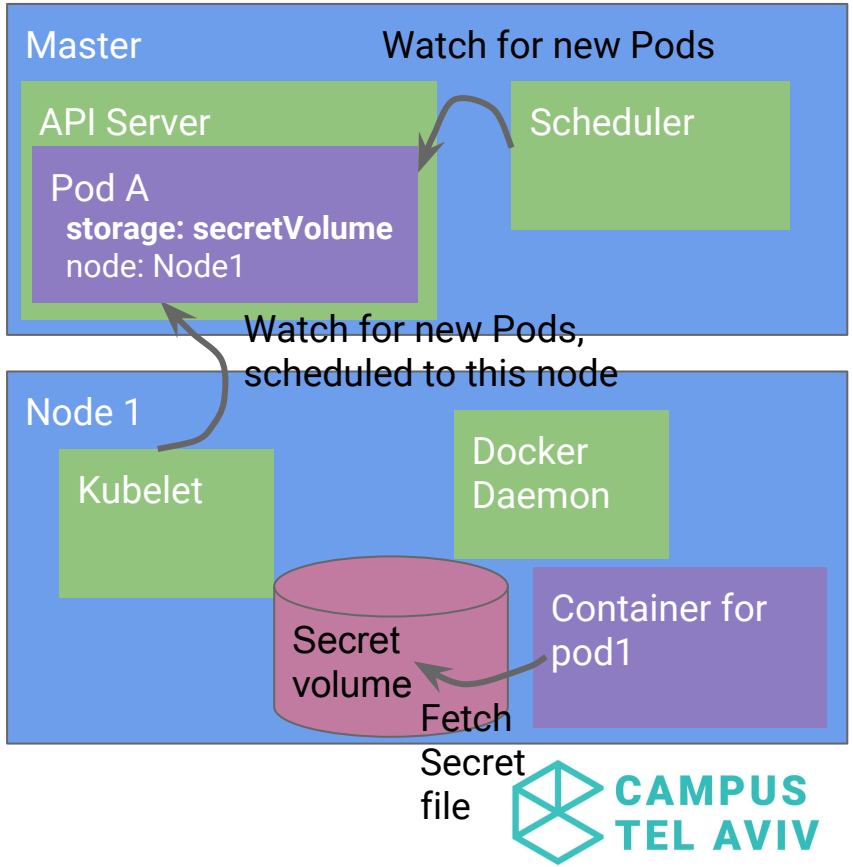
Fetching Kube API Data

- Principle: The control plane should be transparent -- there are no hidden internal APIs.
- Modify application to communicate directly with API Server
- Principle: Meet the user where they are.
- Do not require an app to be re-rewritten to work in Kubernetes.
- Many apps accept secrets and config info as files or env variables, expose Kube API in the way that.



Fetching Kube API Data

- Principle: Meet the user where they are.
- App can consume Secrets, ConfigMaps, and DownwardAPI in the way that it knows how to already (files and env variables).

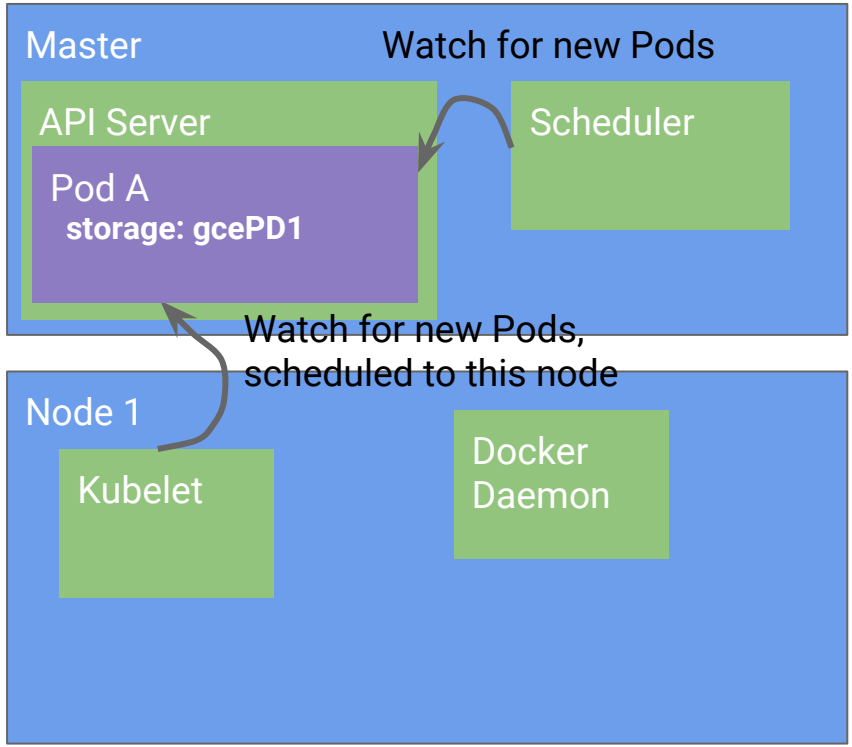


Why meet the user where they are?

Minimize hurdles for deploying workloads on Kubernetes.

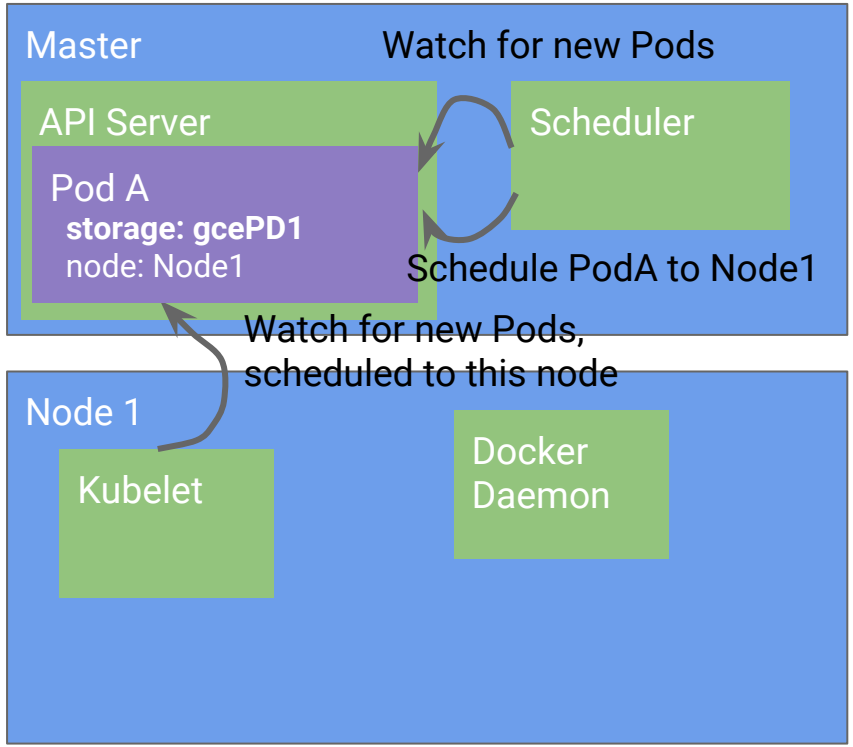
Remote Storage

- Could directly reference a remote volume (GCE PD, AWS EBS, NFS, etc.) in pod definition just like ephemeral volumes (EmptyDir, SecretVolume, etc.).
- Kubernetes will automatically make it available to workload



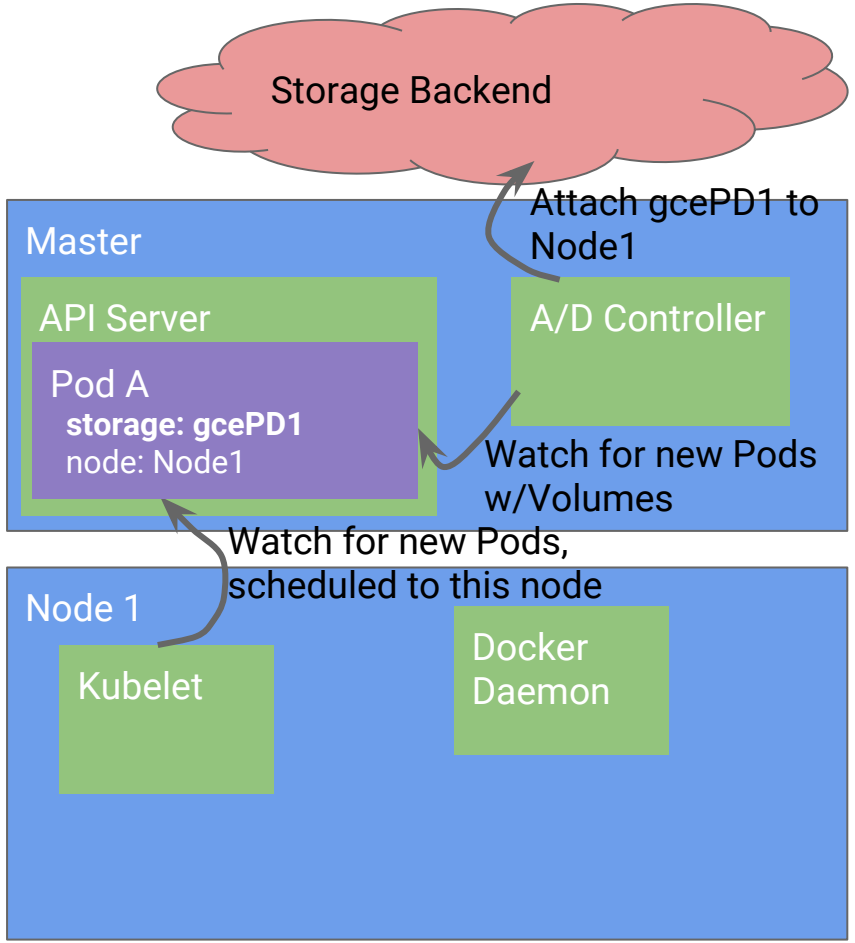
Remote Storage

- Could directly reference a remote volume (GCE PD, AWS EBS, NFS, etc.) in pod definition just like ephemeral volumes (EmptyDir, SecretVolume, etc.).
- Kubernetes will automatically make it available to workload



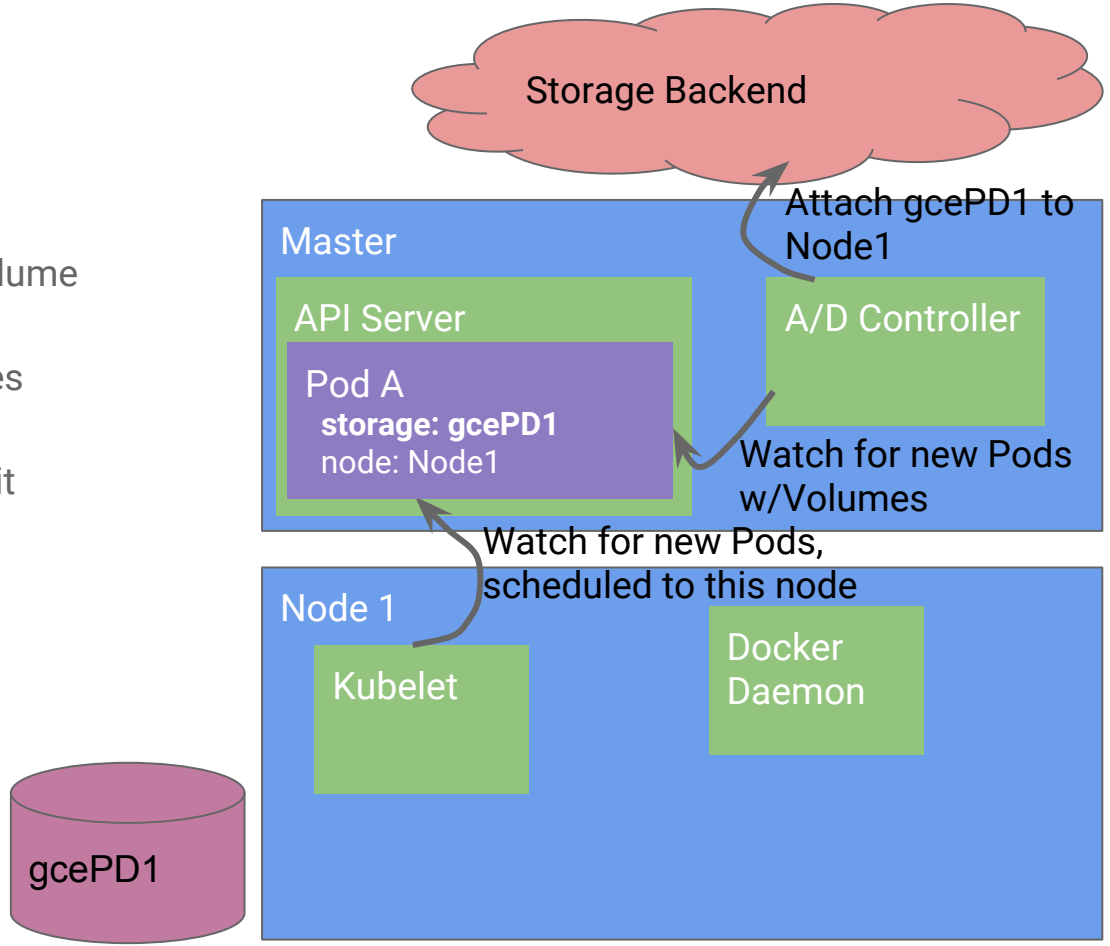
Remote Storage

- Could directly reference a remote volume (GCE PD, AWS EBS, NFS, etc.) in pod definition just like ephemeral volumes (EmptyDir, SecretVolume, etc.).
- Kubernetes will automatically make it available to workload
- Principle: The control plane should be transparent -- there are no hidden internal APIs.



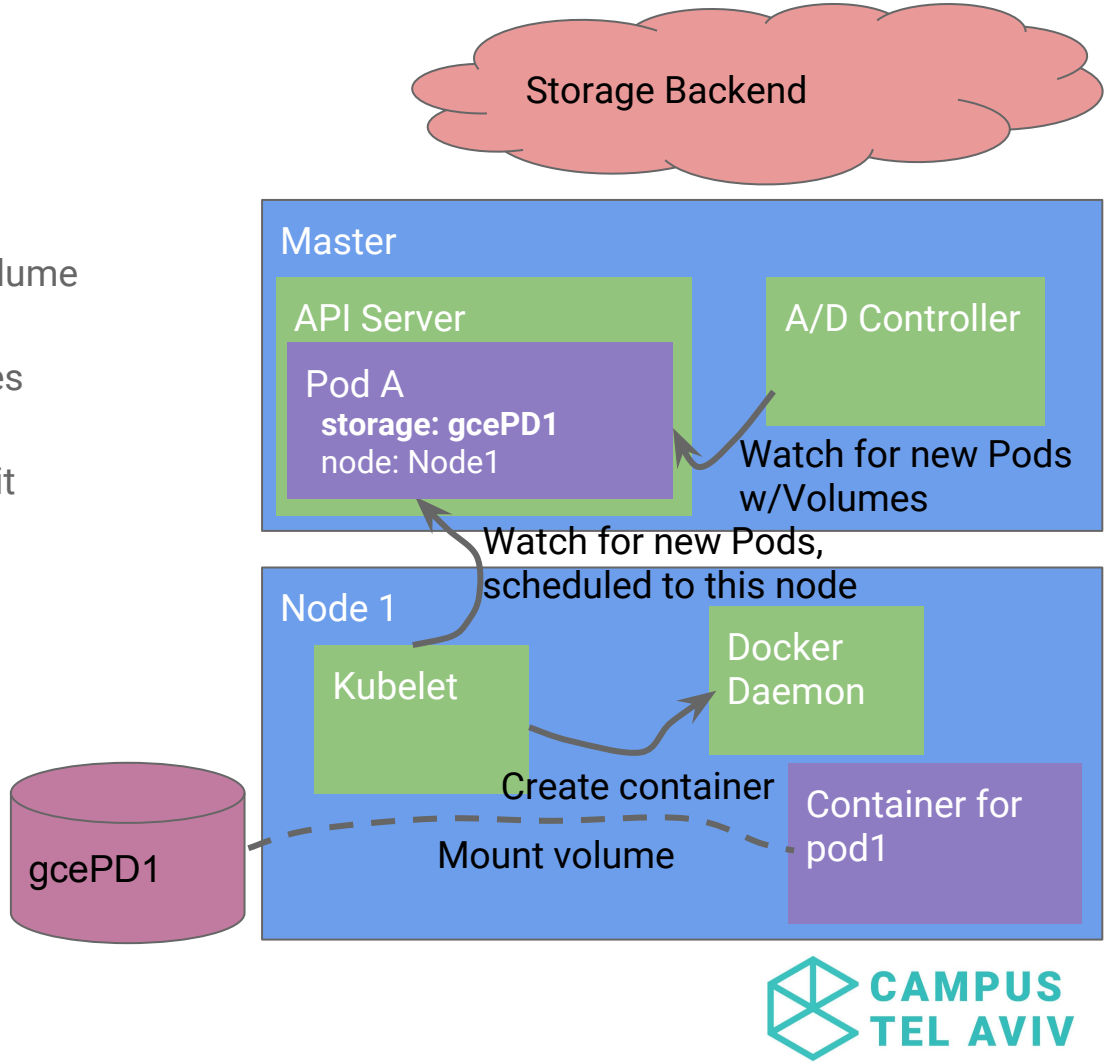
Remote Storage

- Could directly reference a remote volume (GCE PD, AWS EBS, NFS, etc.) in pod definition just like ephemeral volumes (EmptyDir, SecretVolume, etc.).
- Kubernetes will automatically make it available to workload



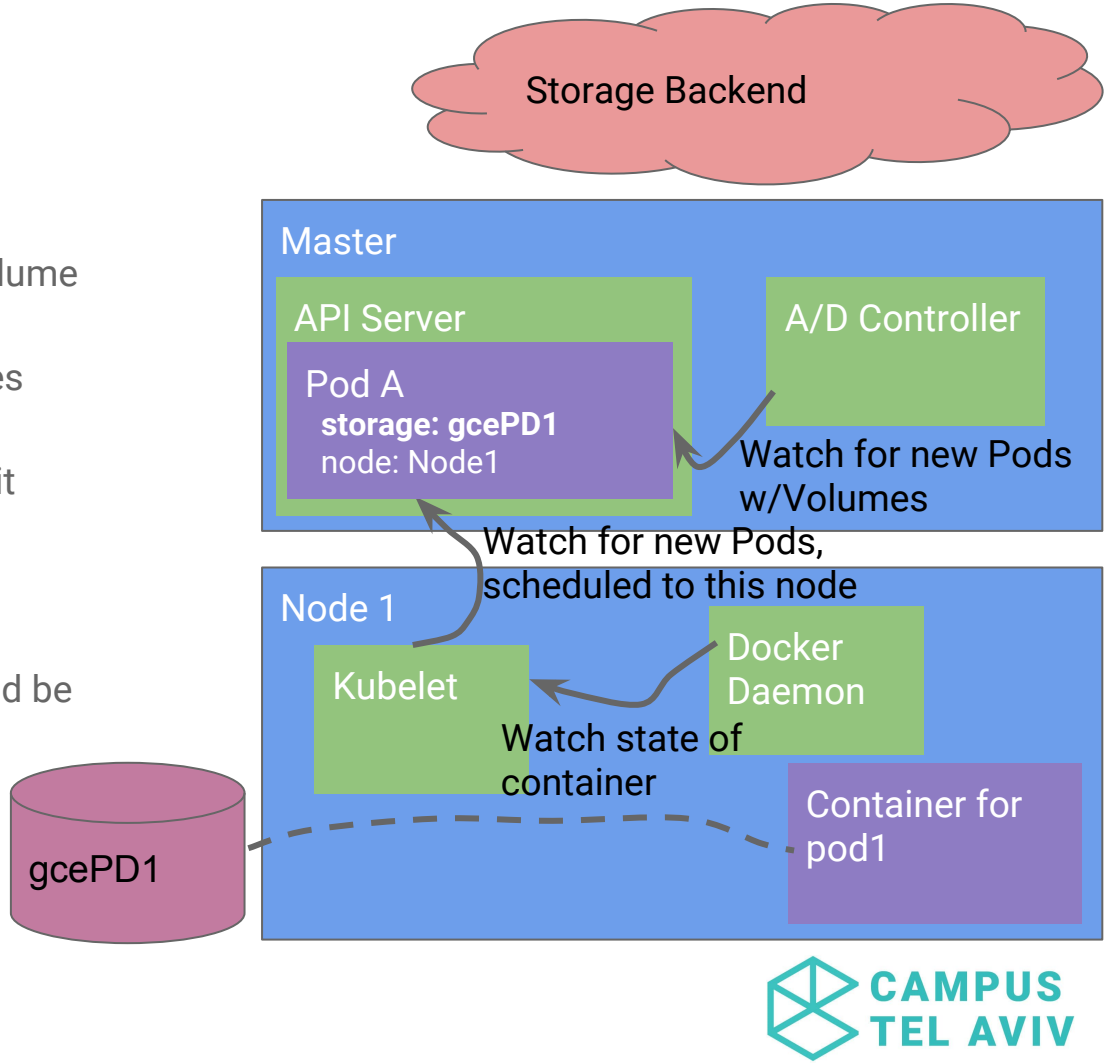
Remote Storage

- Could directly reference a remote volume (GCE PD, AWS EBS, NFS, etc.) in pod definition just like ephemeral volumes (EmptyDir, SecretVolume, etc.).
- Kubernetes will automatically make it available to workload



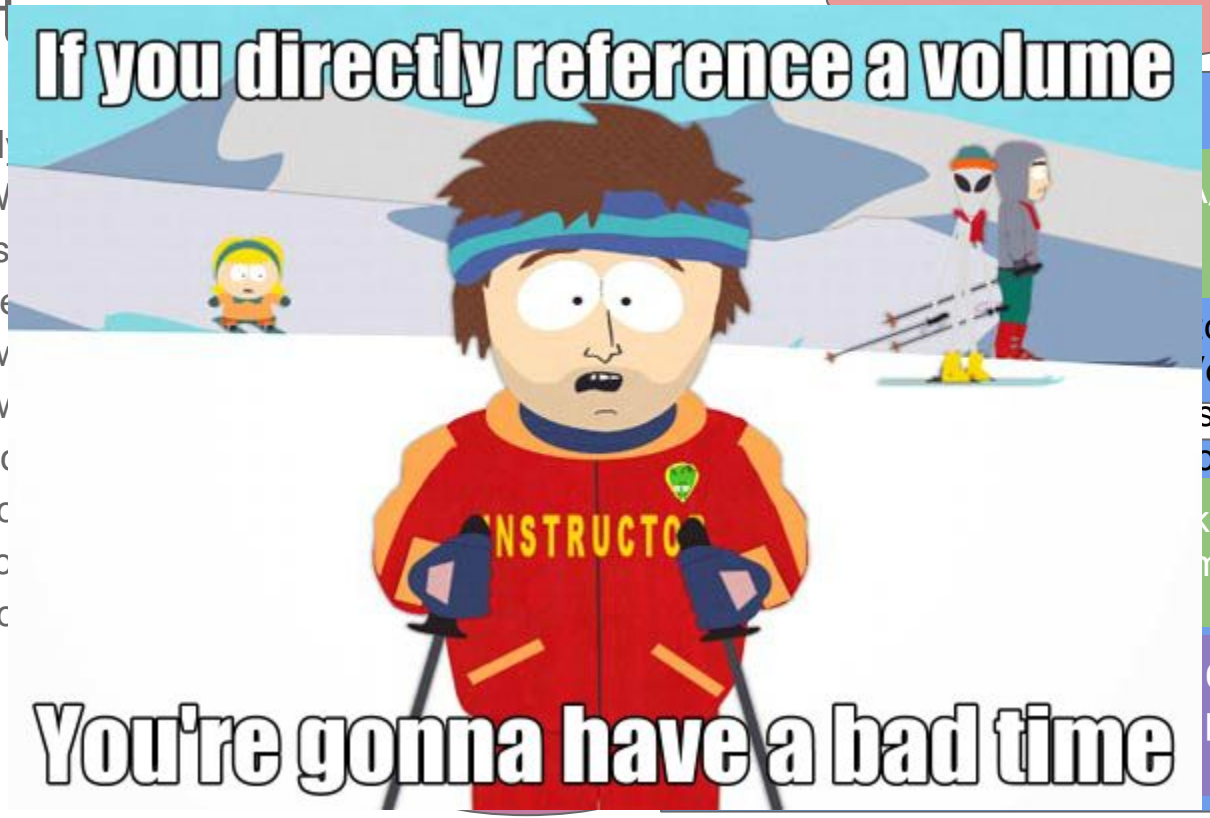
Remote Storage

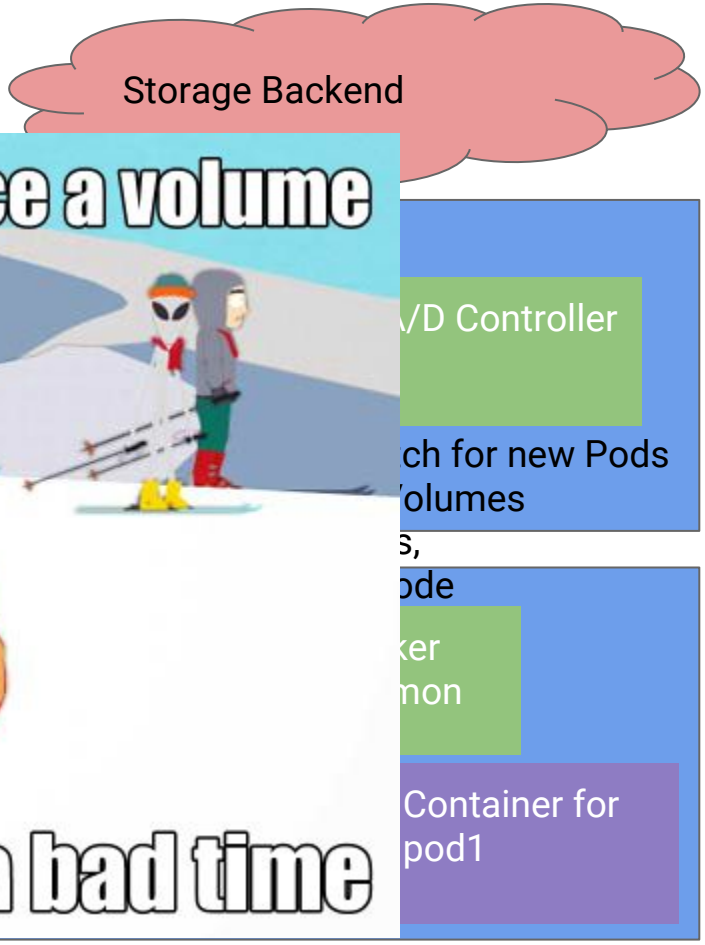
- Could directly reference a remote volume (GCE PD, AWS EBS, NFS, etc.) in pod definition just like ephemeral volumes (EmptyDir, SecretVolume, etc.).
- Kubernetes will automatically make it available to workload
- Problem: Pod definition is no longer portable across clusters.
- Principle: Workload definitions should be portable across clusters.



Remote St

- Could directl
  (GCE PD, AW
  definition jus
  (EmptyDir, Se
- Kubernetes v
  available to v
- Problem: Poc
  portable acrc
- Principle: Wc
  portable acrc





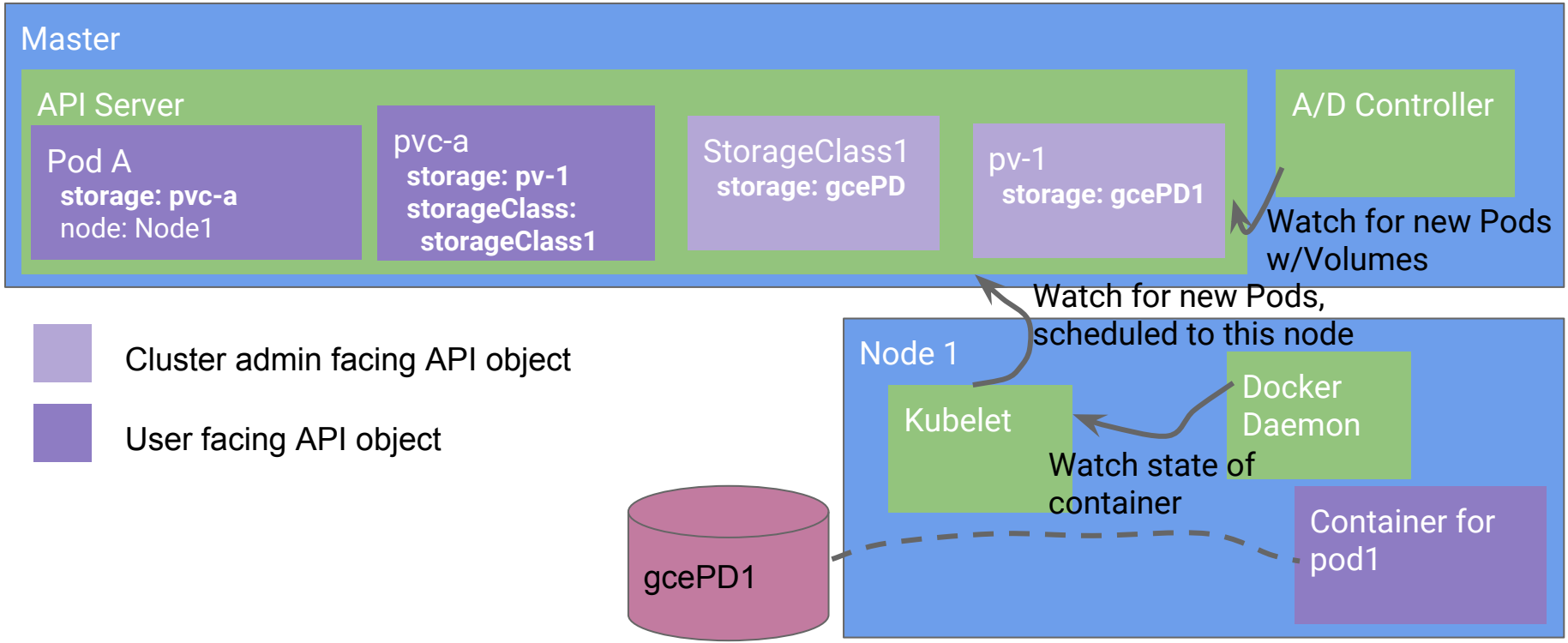
Google

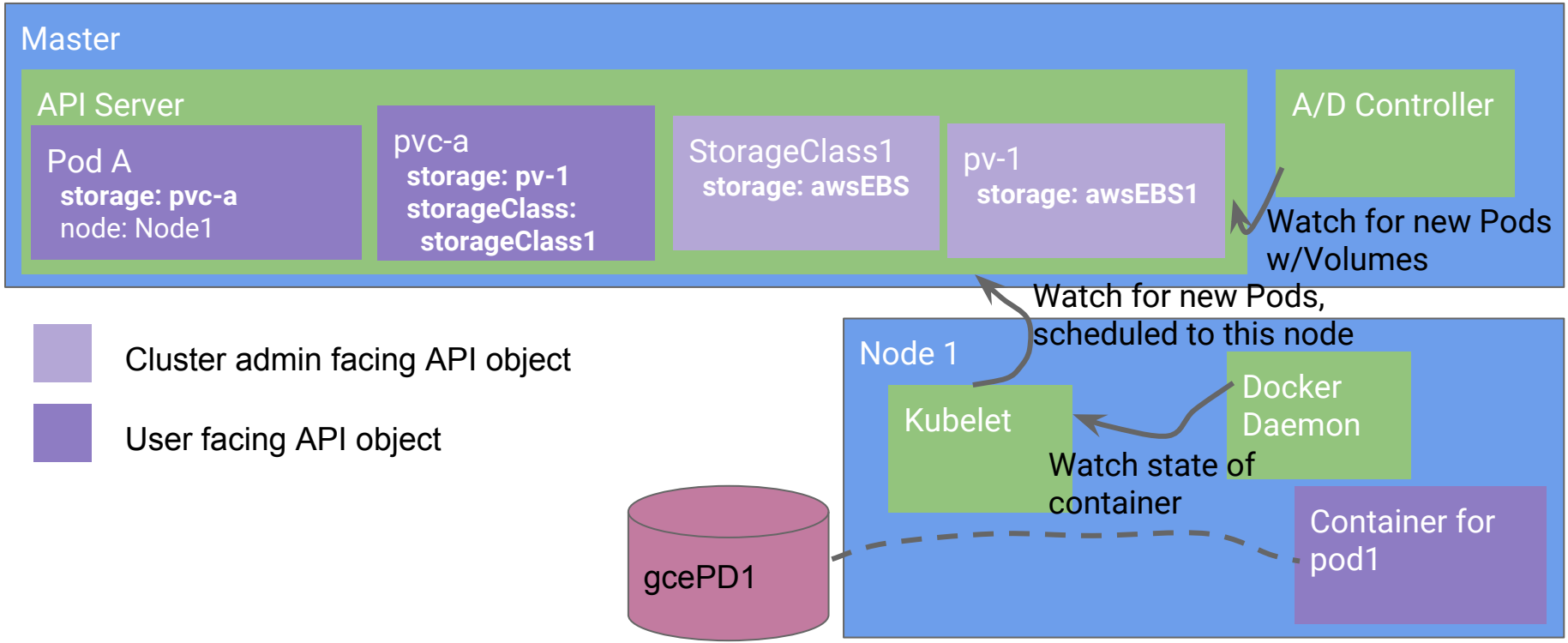
CAMPUS TEL AVIV

PV/PVC

PersistentVolume and PersistentVolumeClaim Abstraction

Decouple storage implementation from storage consumption





Why workload portability?

Decouple distributed system application development from cluster implementation.
Make Kubernetes a true abstraction layer, like an OS.

Kubernetes Principles Introduced

1. Kube API declarative over imperative.
2. No hidden internal APIs
3. Meet the user where they are
4. Workload portability

Question?

Get Involved!

- Kubernetes Storage Special-Interest-Group (SIG)
  - github.com/kubernetes/community/tree/master/sig-storage
  - Meeting every 2 weeks, Thursdays at 9 AM PST (7 PM IT)
- Mailing list:
  - kubernetes-sig-storage@googlegroups.com
- Contact me:
  - Saad Ali, Google
  - github.com/saad-ali
  - twitter.com/the_saad_ali