Airflow on Kubernetes: Containerizing your Workflows

By Michael Hewitt

Agenda

1. Kubernetes Overview
2. Airflows integration with Kubernetes
3. Deployment of Airflow on Kubernetes
4. Kubernetes Pod Operator and its benefits
5. DAG Development Transformations
6. The Future of Airflow on Kubernetes

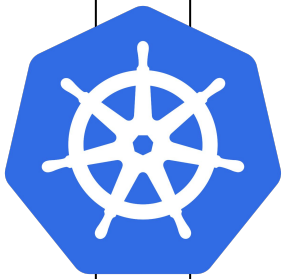
Kubernetes

Scalable

- Horizontally scaling infrastructure
- Automated scaling of containers based on system level metrics
- Manual scaling of containers
- Components that keep track of application replicas, scale in and out as needed

Extensible

- Supports configuration to schedule containers on certain types nodes automatically
- Supports the use of multiple schedulers at the same time
- Dynamic Webhook



Highly Available

- Easily integrate health checks
- Self healing containers
- Native load balancers to automatically divert container traffic
- Automated scaling based on L7 metrics

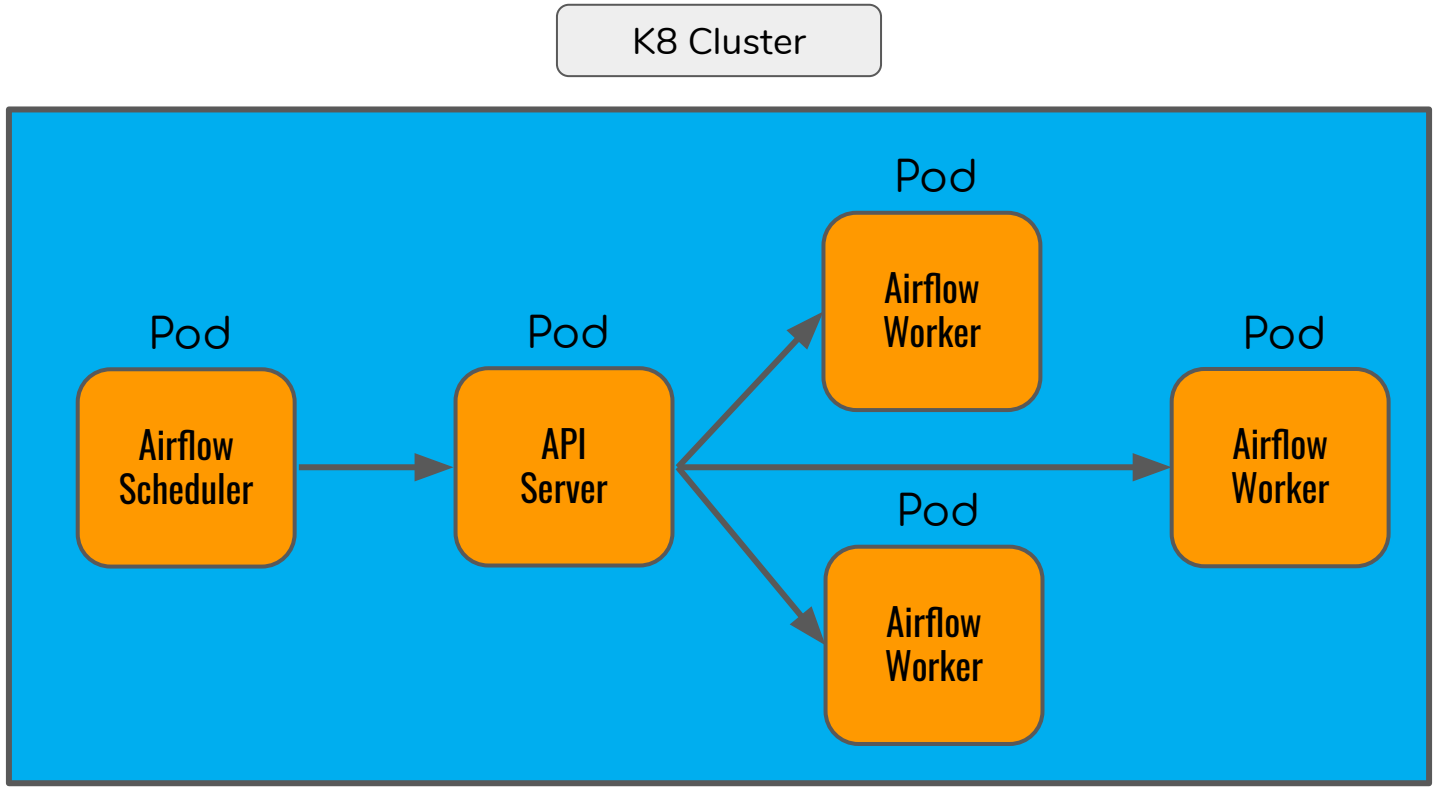
Usability

- Supports both declarative and imperative configuration
- Supports APIs for a plethora of languages
- Usable executor for other platforms (Airflow, Gitlab)

The Pod

- A Pod is the basic execution unit of a Kubernetes application
- Abstraction of a container or group of containers representing a process
- Easily expose the containers within pods
- Each pod has its own network namespace making containers within the same pod reachable by localhost
- Supports both ephemeral storage and persistent storage that can easily be shared between pods/containers

Kubernetes Executor



Kubernetes Executor Benefits

Dynamic amount of workers unlike other executors

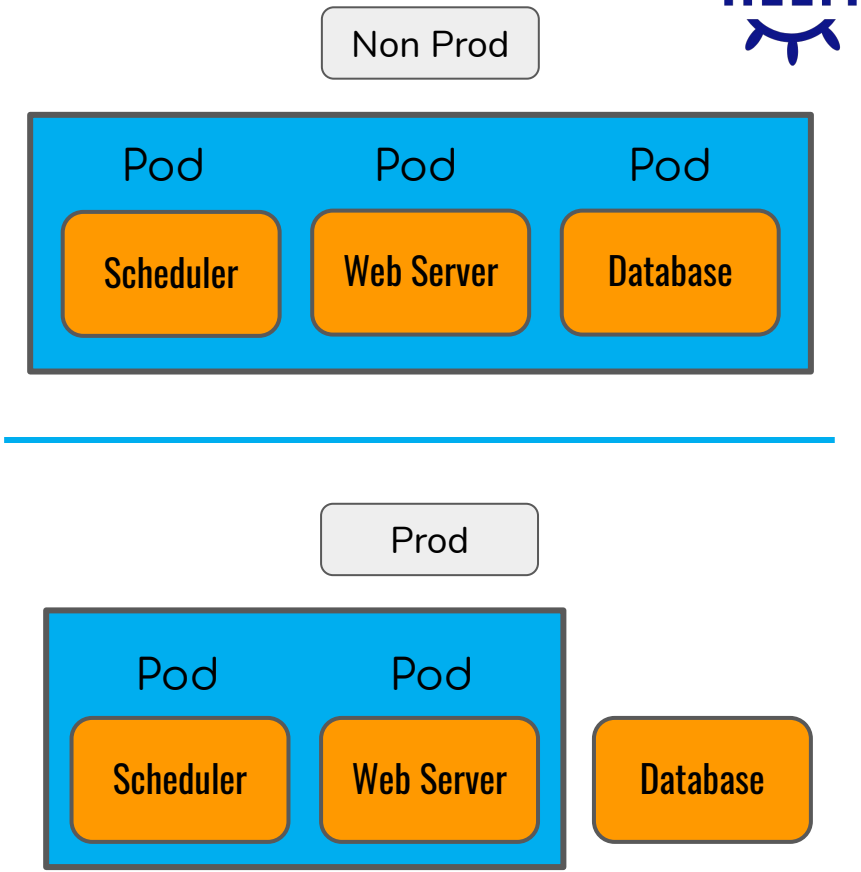
Avoids wasted resources

Fault tolerance as tasks are now isolated in pods

Reduced stress on Airflow Scheduler due to edge-driven triggers in K8S Watch API

Deploy Airflow with Helm

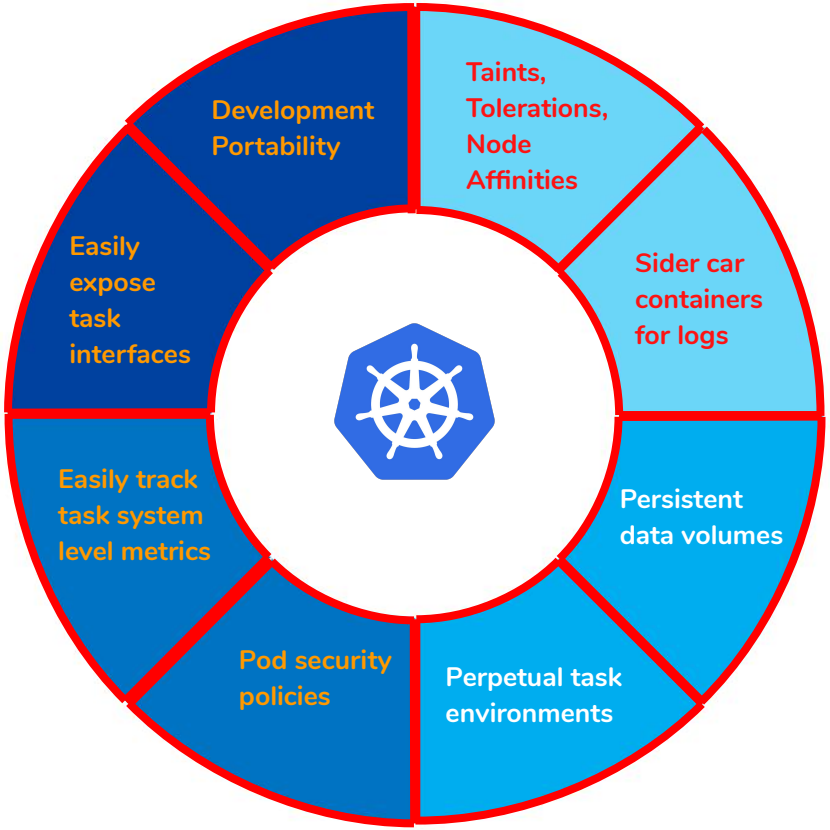
- Package manager for Kubernetes
- Deploy and manage multiple manifests as one unit
- Golang templating language to templatize manifests
- Automate deployment of Airflow with Helm using Terraform



Kubernetes Pod Operator

```
passing = KubernetesPodOperator(namespace='default',
                                image="python:3.6",
                                cmds=["python","-c"],
                                arguments=["print('hello world')"],
                                labels={"foo": "bar"},
                                name="passing-test",
                                task_id="passing-task",
                                get_logs=True,
                                dag=dag
                                )
```

Take Control with Kubernetes



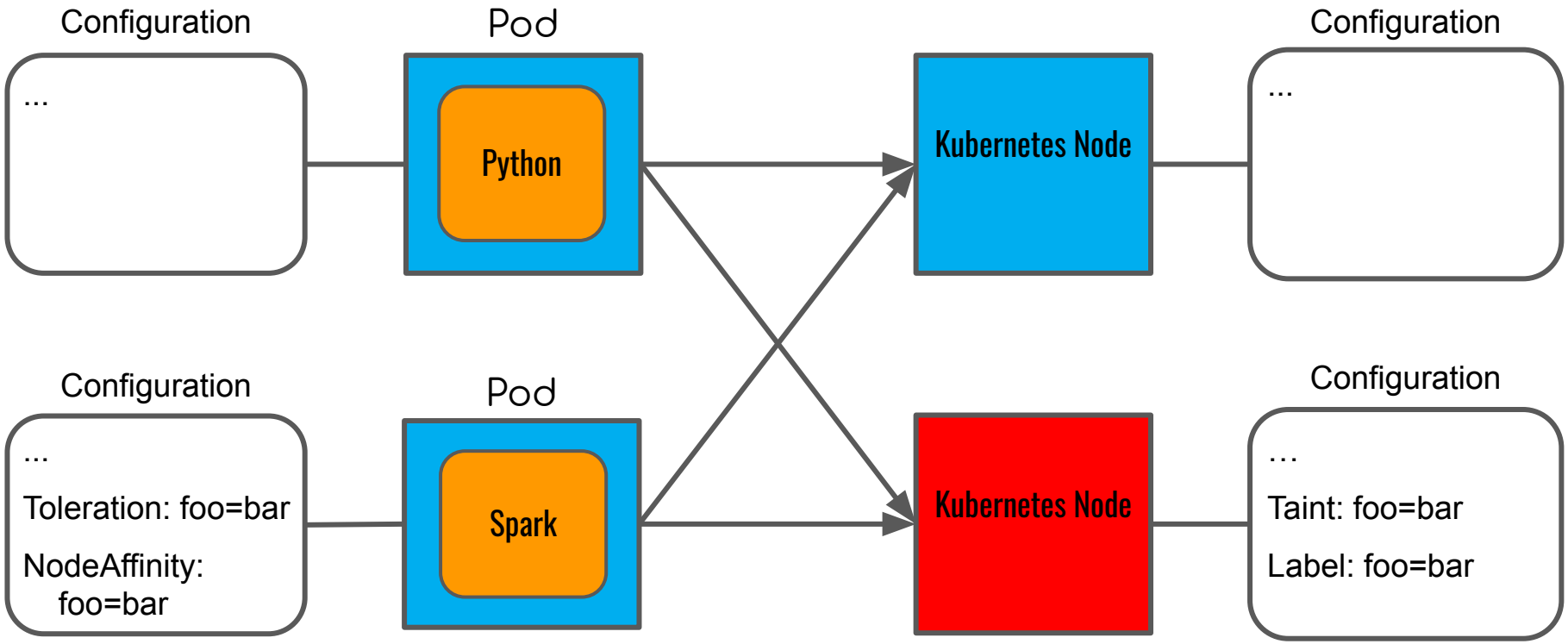
Executor Config

```
traf_type_sensor = S3KeySensor(
        task_id = traf_type + '-' + str(30*index) + 'minutes',
        soft_fail = False,
        mode = 'reschedule',
        bucket_key = traf_type_s3key_tmpl,
        bucket_name = s3_sensor_bucket_name,
        aws_conn_id='aws_default',
        on_failure_callback=callback,
        executor_config=
                        {
                            "KubernetesExecutor":
                            {
                                "annotations":
                                {
                                    "iam.amazonaws.com/role":iam_role
                                }
                            }
                        },
        dag = dag)
```

Adapting DAG Development

- Airflow configuration with Kubernetes
- Kubernetes RBAC
- IAM roles/policies
- Automate with Terraform
  - K8S resources
  - IAM role/policies
  - Pod Networking policies
  - Datadog dashboard for alerts and metrics
- Template environments with CI/CD

Taints, Tolerations, and Node Affinities



Abstracting Kubernetes through Webhooks

- Some K8S concepts have sharp learning curves
- SREs typically manage the Kubernetes clusters
- Dynamic Webhook
  - Validating Webhooks enable an extra validation on K8S API calls
  - Mutating Webhook enable the automatic addition of properties on K8S resource creation
- Developer apply labels(simple concept) mutating webhook applies toleration and Affinities
- Force teams to label pods with team name, cost center, etc., with validating webhooks

What’s Next: Airflow 2.0

- Directly apply pod manifests in Kubernetes Pod Operator
- Kubernetes Spark Operator
- New Official Airflow Docker Image
- New Official Airflow Helm Chart