Exploring and Verifying the Output

In this lesson, we will explore the sequential occurrence of events as a result of the rolling update and verify the change of Kubernetes version.

Exploring the Sequence of Events

So the rolling update finished and the **output** starts with the same information we got when we asked for a preview, so there's not much to comment.

```
I0225 23:03:03.993068       1 instancegroups.go:130] Draining the node: "ip-172-20-40-167..."
node "ip-172-20-40-167..." cordoned
node "ip-172-20-40-167..." cordoned
WARNING: Deleting pods not managed by ReplicationController, ReplicaSet, Job, DaemonSet or St
node "ip-172-20-40-167..." drained
```

Instead of destroying the first node, kops picked one master and drained it. That way, the applications running on it can shut down gracefully. We can see that it drained:

- `etcd-server-events`
- `etcd-server-ip`
- `kube-apiserver`
- `kube-controller-manager`
- `kube-proxy`
- `kube-scheduler`
- Pods running on the server `ip-172-20-40-167`

As a result, Kubernetes rescheduled the Pods to one of the healthy nodes. That might not be true for all the Pods but only for those that can be rescheduled.

```
I0225 23:04:37.479407 1 instancegroups.go:237] Stopping instance "i-06d40d6ff583fe10b", node
```

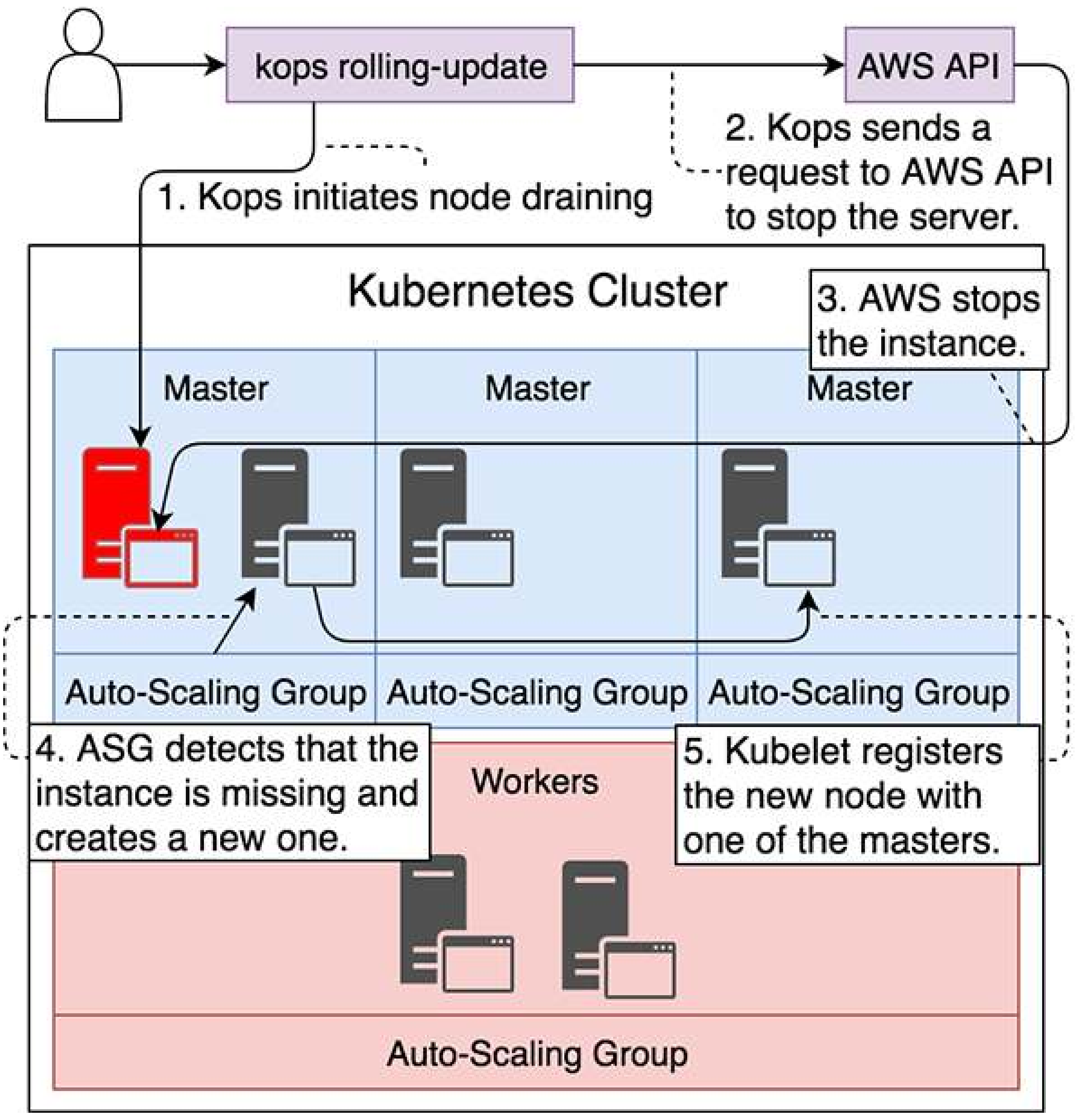
We can see that after draining finished, the master node was stopped. Since each master is associated with an auto-scaling group, AWS will detect that the node is no more, and start a new one.

Once the new server is initialized, `nodeup` will execute and install Docker, Kubelet, and Protokube. The latter will create the manifest that will be used by Kubelet to run the Pods required for a master node. Kubelet will also register the new node with one of the healthy masters.

That part of the process is the same as the one executed when creating a new cluster or when adding new servers. It is the part that takes longest to complete (around five minutes).

```
I0225 23:09:38.218945 1 instancegroups.go:161] Validating the cluster.
I0225 23:09:39.437456 1 instancegroups.go:212] Cluster validated.
```

We can see that, after waiting for everything to settle, kops validated the cluster, thus confirming that upgrade of the first master node finished successfully.


kops rolling-update
AWS API
1. Kops initiates node draining
2. Kops sends a request to AWS API to stop the server.
Kubernetes Cluster
3. AWS stops the instance.
Master
Master
Master
Auto-Scaling Group
Auto-Scaling Group
Auto-Scaling Group
4. ASG detects that the instance is missing and creates a new one.
Workers
5. Kubelet registers the new node with one of the masters.
Auto-Scaling Group

Rolling upgrade of one of the master nodes

As soon as it validated the upgrade of the first master, kops proceeded with the next node. During next ten to fifteen minutes, the same process will be repeated with the other two masters.

Once all three are upgraded, kops will execute the same process with the worker nodes, and we'll have to wait for another ten to fifteen minutes.

```
I0225 23:34:01.148318 1 rollingupdate.go:191] Rolling update completed for cluster "devops23.
```

Finally, once all the servers were upgraded, we can see that rolling update was completed.

Verification

Let's go back to our cluster and verify that Kubernetes was indeed upgraded.

```
kubectl get nodes
```

The **output** is as follows.

```
NAME                  STATUS ROLES  AGE VERSION
ip-172-20-107-172... Ready  node   4m  v1.9.2
ip-172-20-124-177... Ready  master 16m v1.9.2
ip-172-20-44-126...  Ready  master 28m v1.9.2
ip-172-20-56-244...  Ready  node   10m v1.9.2
ip-172-20-67-40...   Ready  master 22m v1.9.2
```

Judging by versions of each of the nodes, all were upgraded to `v1.9.2`. The process worked.

> Try to upgrade often. As a rule of thumb, you should upgrade one minor release at a time.

Even if you are a couple of minor releases behind the stable kops-recommended release, it's better if you execute multiple rolling upgrades (one for each minor release) than to jump to the latest at once. By upgrading to the next minor release, you'll minimize potential problems and simplify rollback if required.

Even though kops is fairly reliable, you should not trust it blindly. It's relatively easy to create a small testing cluster running the same release as production, execute the upgrade process, and validate that everything works as expected. Once finished, you can destroy the test cluster and avoid unnecessary expenses.

> Don't trust anyone. Test upgrades in a separate cluster.

---

n the next lesson, we will explore how to upgrade the cluster automatically.