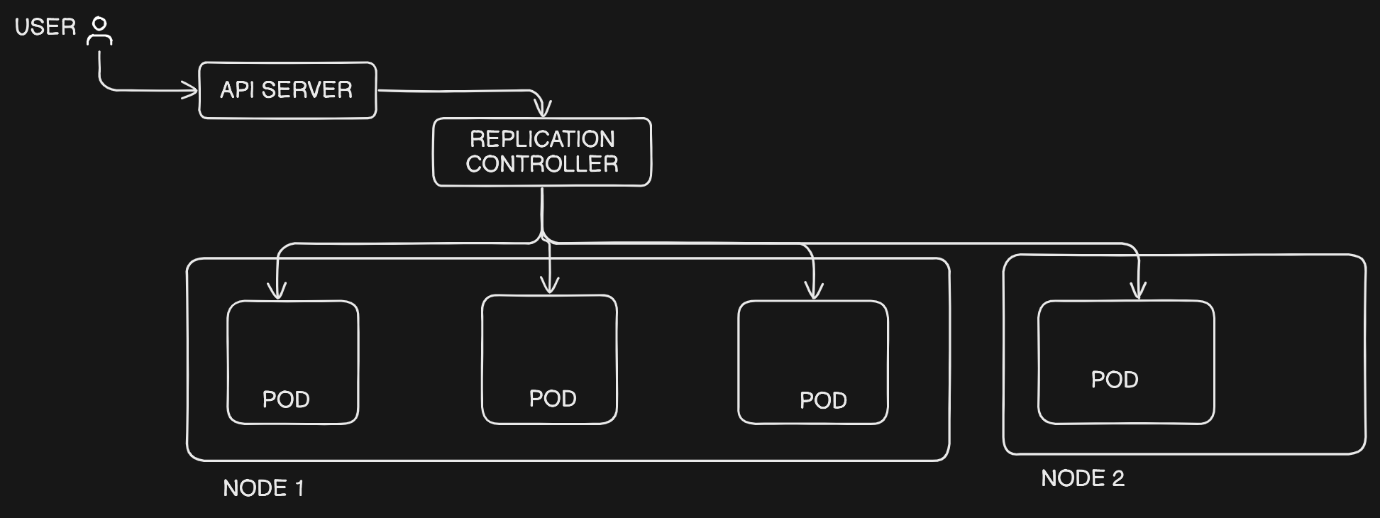
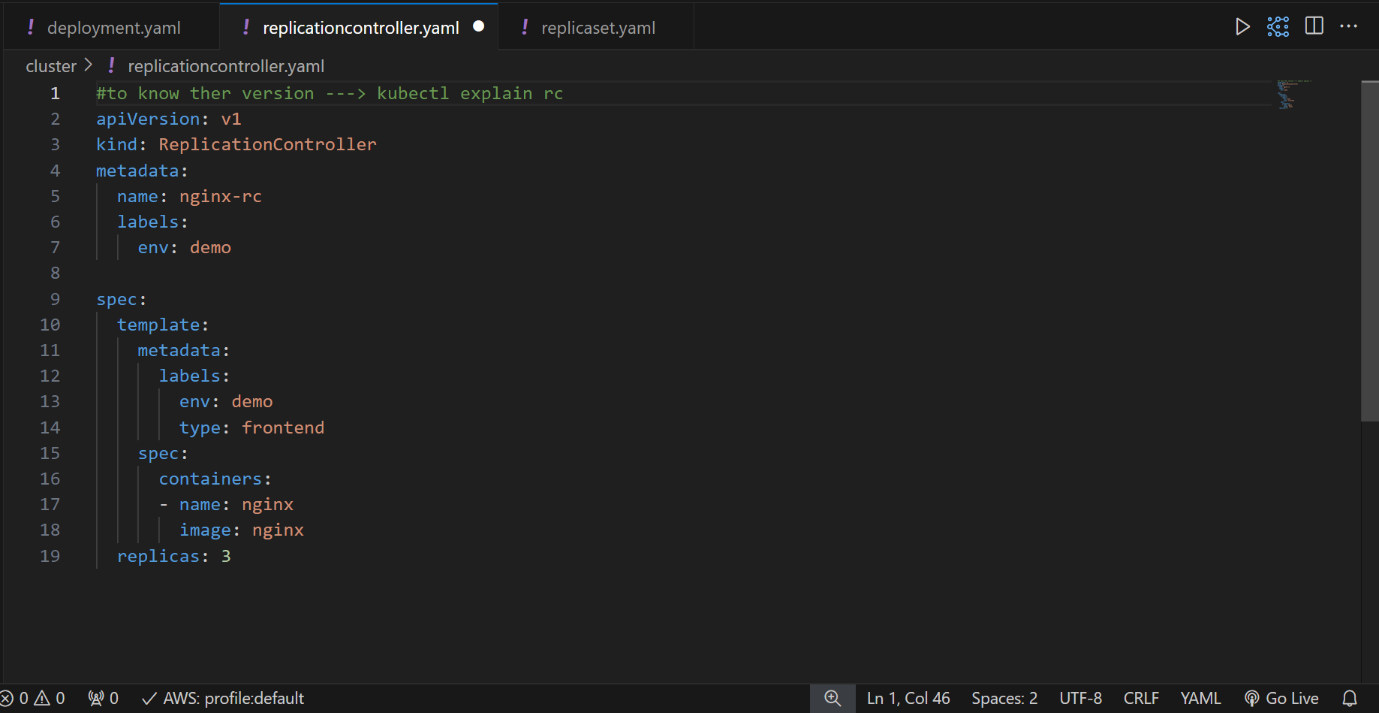
In Kubernetes, **Replication Controller**, **ReplicaSet**, and **Deployment** are all used for ensuring that a certain number of pod replicas are running at any given time.

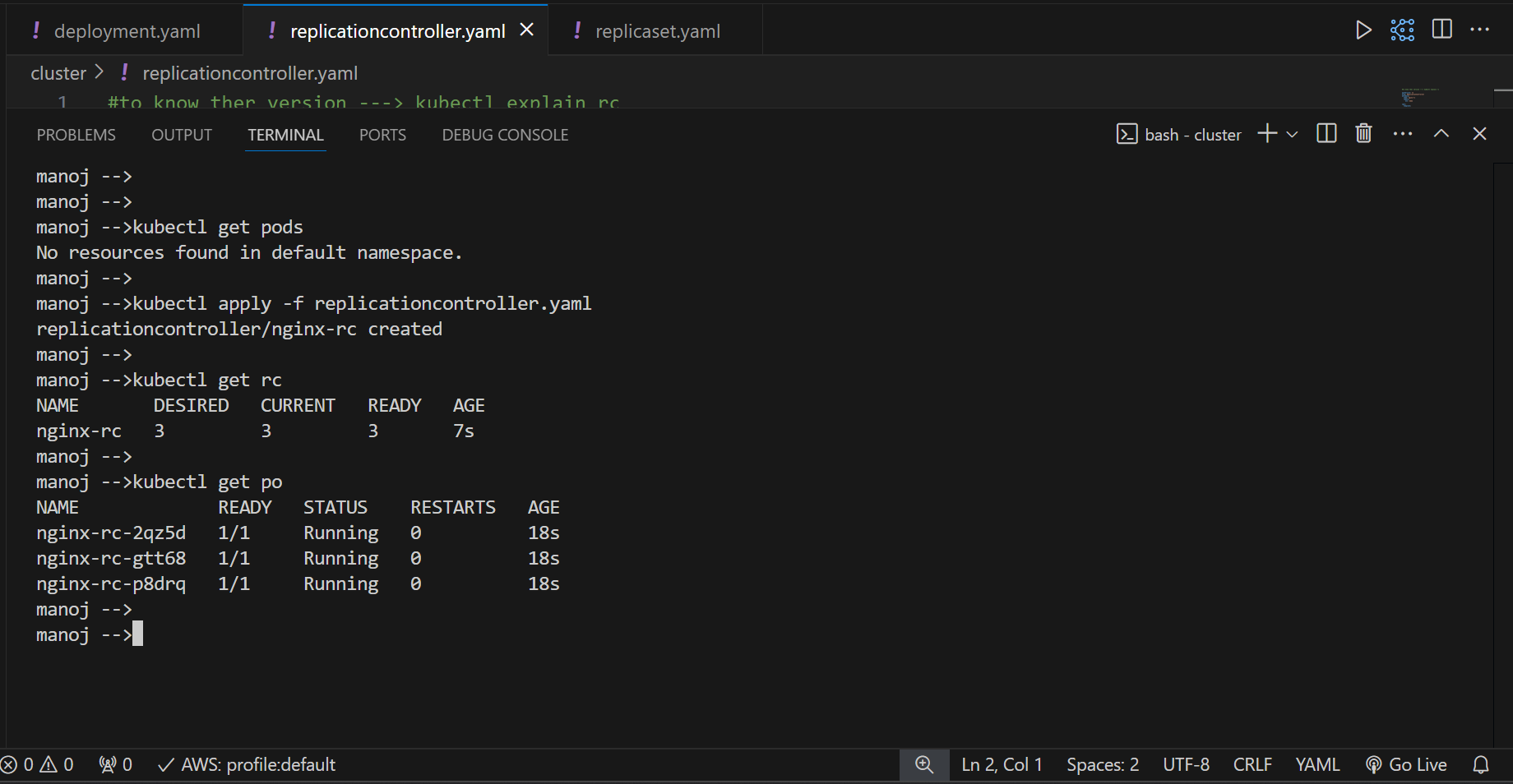
**REPLICATION CONTROLLER (RC)**

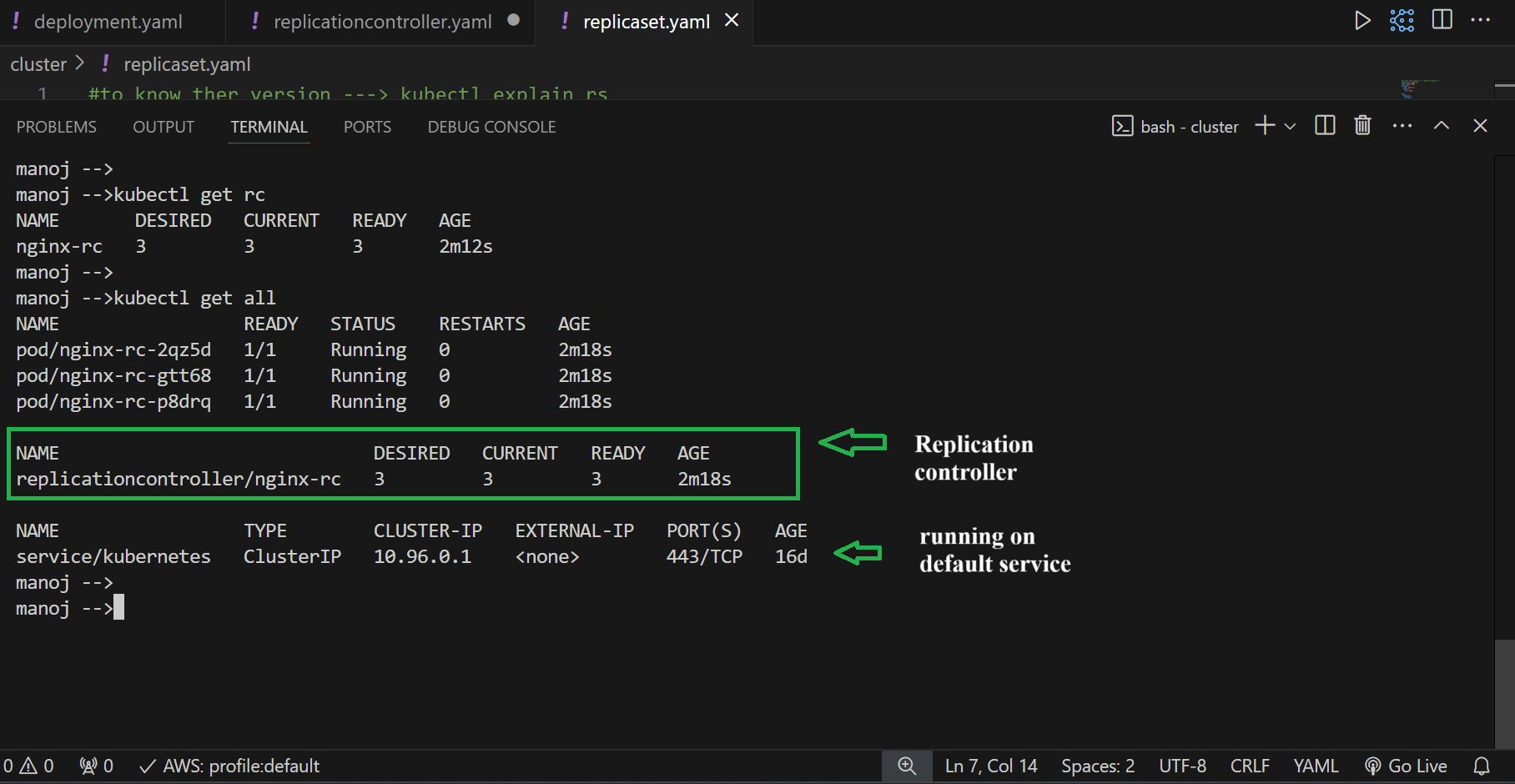


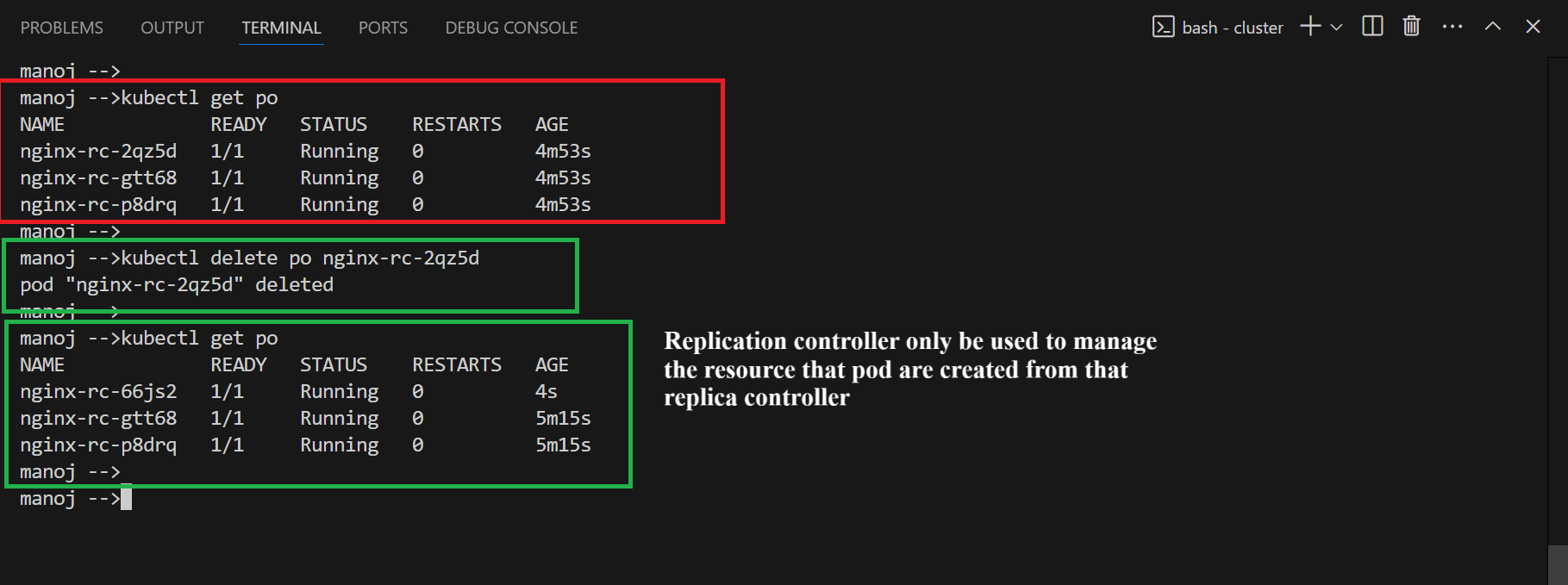
User is trying to access the API, the request will go to API server first and request will be forwarded to replication controller. This controller will choose the POD to which the request should be sent, basically it will choose the healthy POD.

In case the POD goes down replication controller will create another identical POD before it goes down and send the request to that. Let’s a node run out of resource [eg: CPU, memory etc] in that case we can spin new node and re-direct that traffic to new POD at new node.







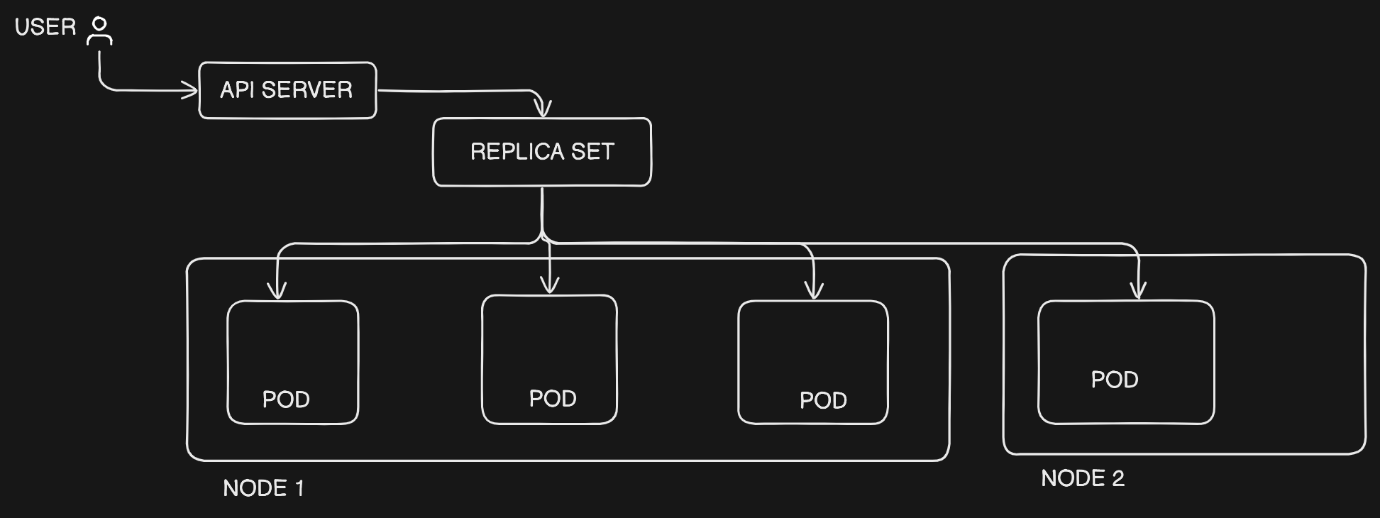


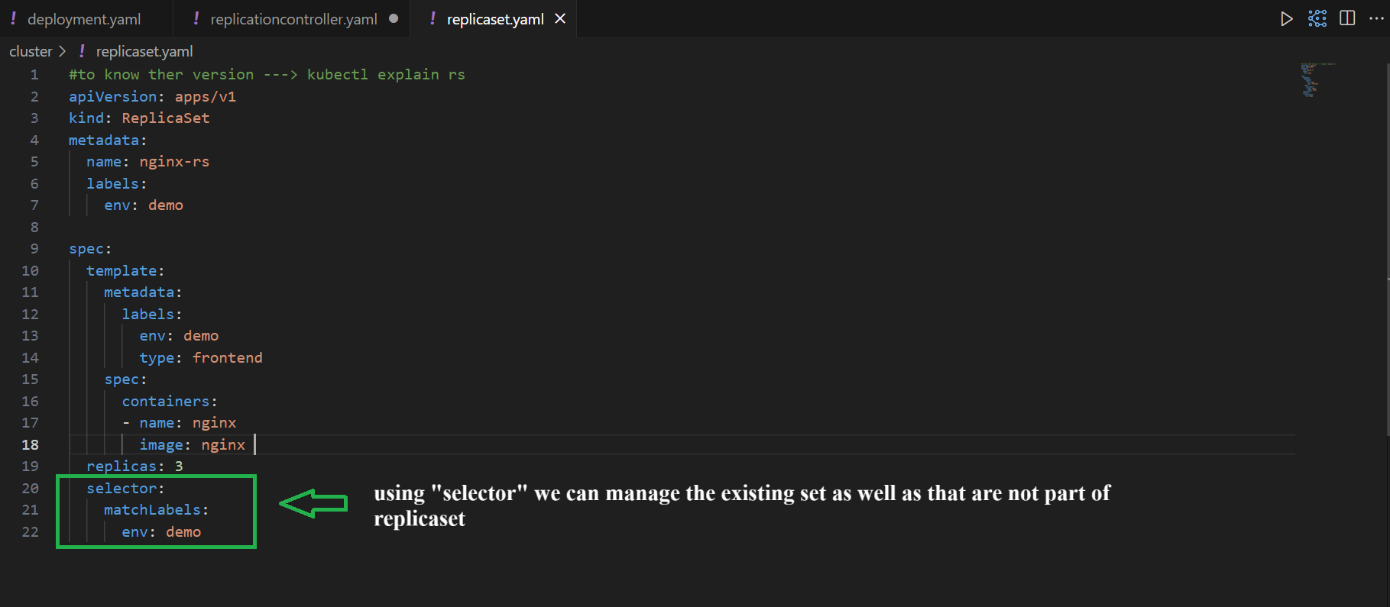
Key points:

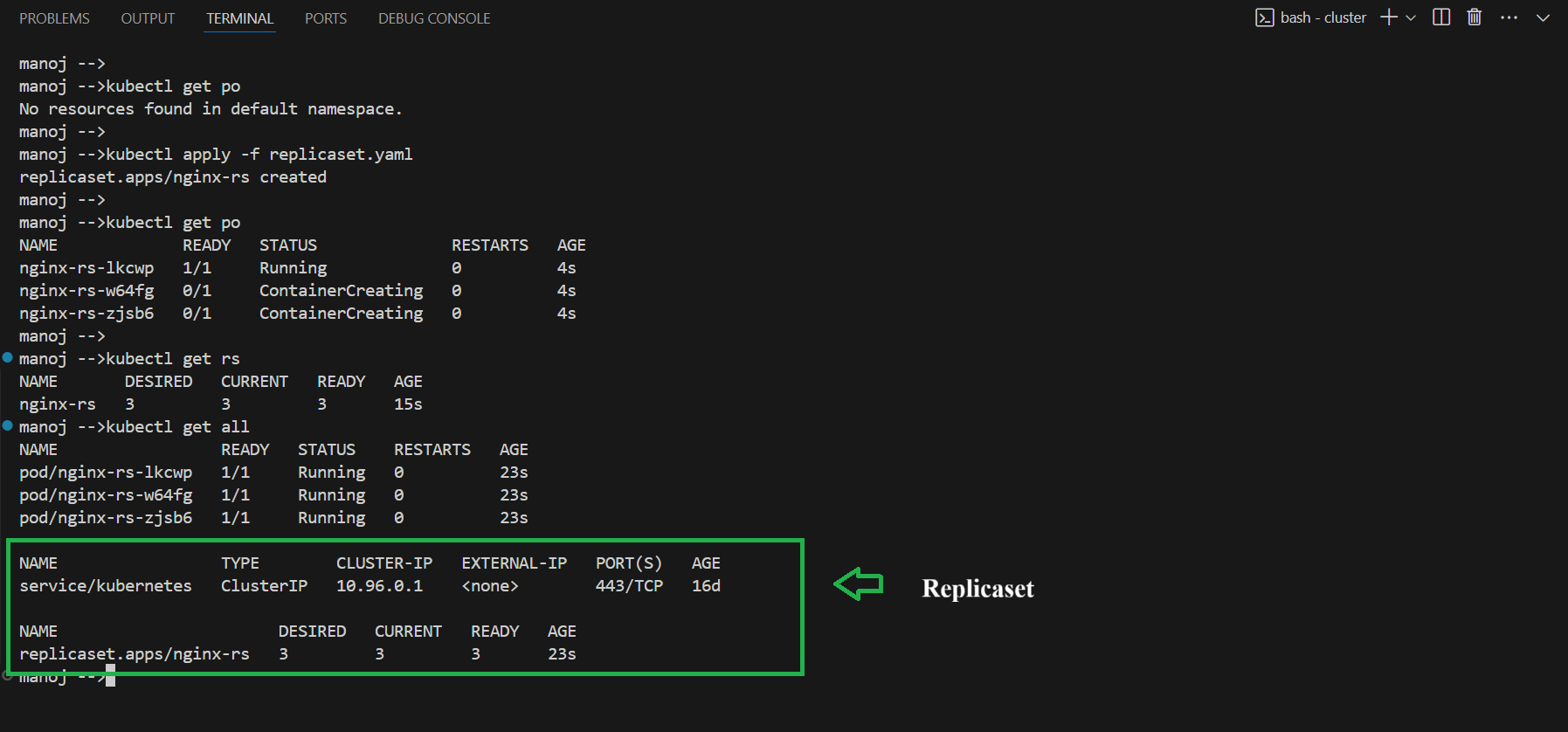
* Ensures a specific number of replicas of a pod are running at any time.
* Does not support advanced features like rolling updates or rollbacks.
* Deprecated in favor of ReplicaSet and Deployment but is still available for backward compatibility.

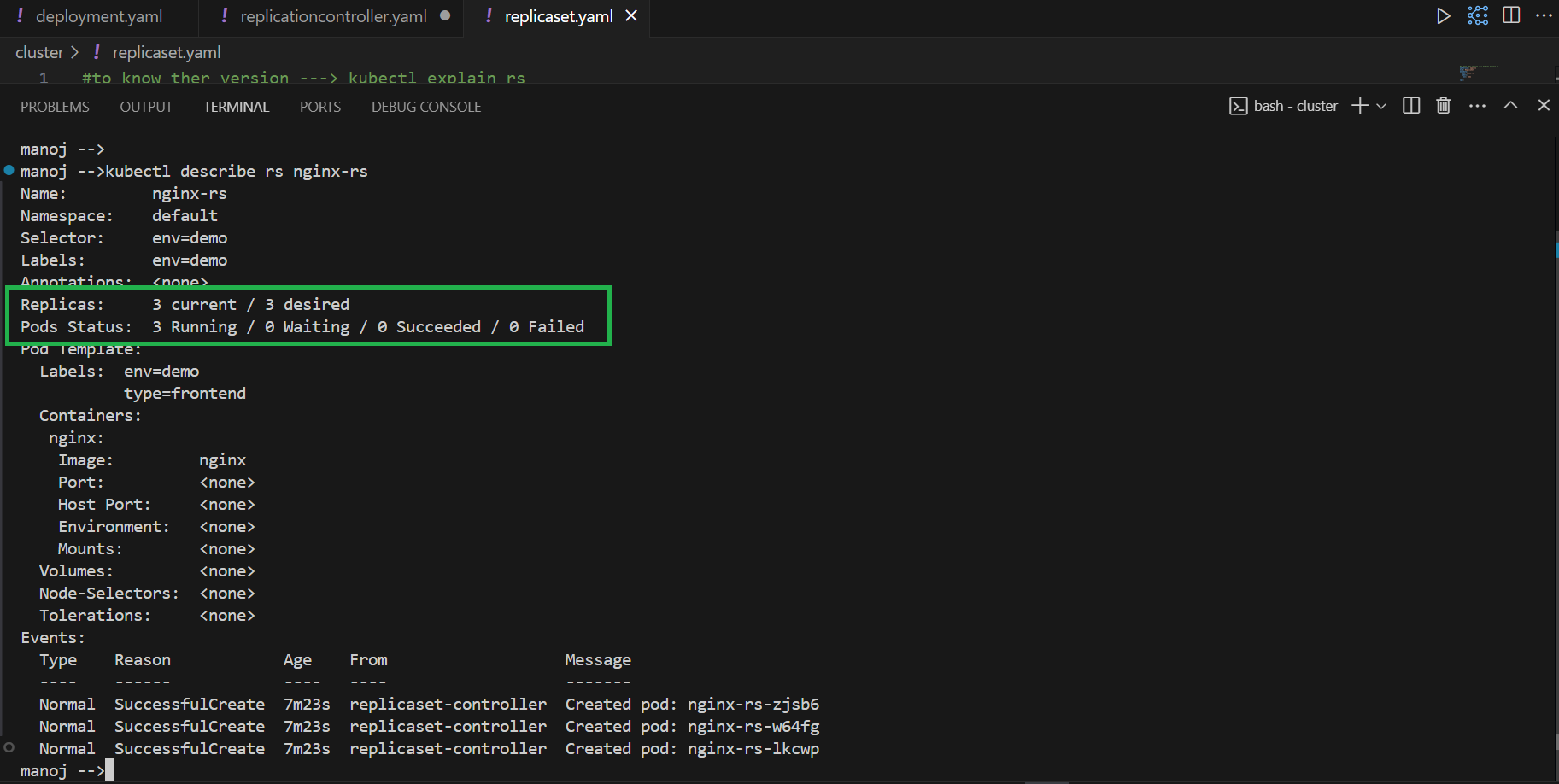
**REPLICASET (RS)**

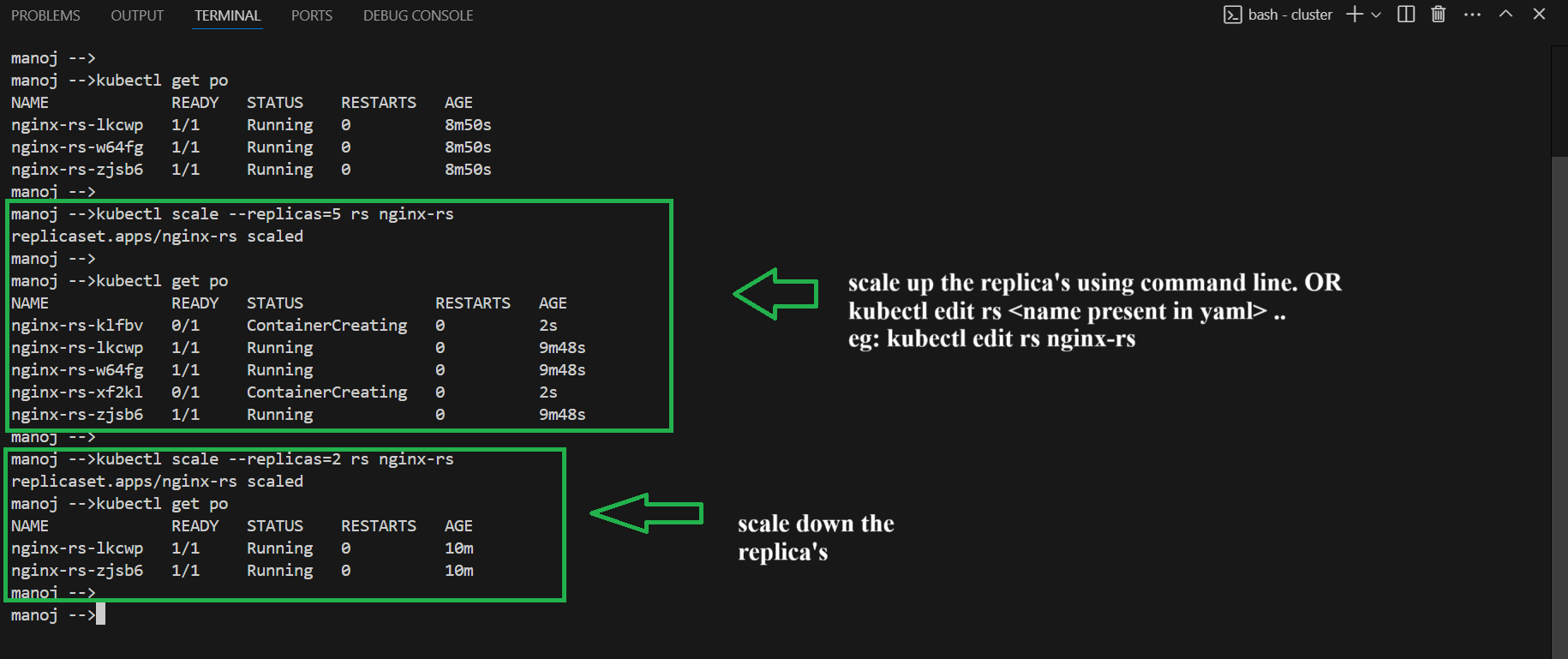
ReplicaSet is the successor to the Replication Controller. It has a similar role but with more features and flexibility, like support for set-based label selectors.











NOTE:

We will use replica set over replication controller because replication controller only be used to manage the resource that pod are created from that replica controller.

Where as in replica set we can manage the existing set as well that are not part of the replica set with the help of “selector” with the “matchLabels”

Key points:

* Ensures a specific number of pod replicas are running.
* Supports set-based label selectors for more flexible pod matching.
* Primarily used by Deployment to control pod scaling, but can also be used on its own.