ONAP IM discussion

Ericsson

Two CNF onboarding and orchestration options

ETSI MANO NFV VNFD Sol001
IFA 11 based

Standardized in ETSI NFV

- ETSI started assuming infrastructure capability based upon VM capabilities.
- The VNFD was created to allow the VNFM to support the necessary lifecycle capabilities.
- Support for CNF was added on top but inheriting the structure from VM based deployments

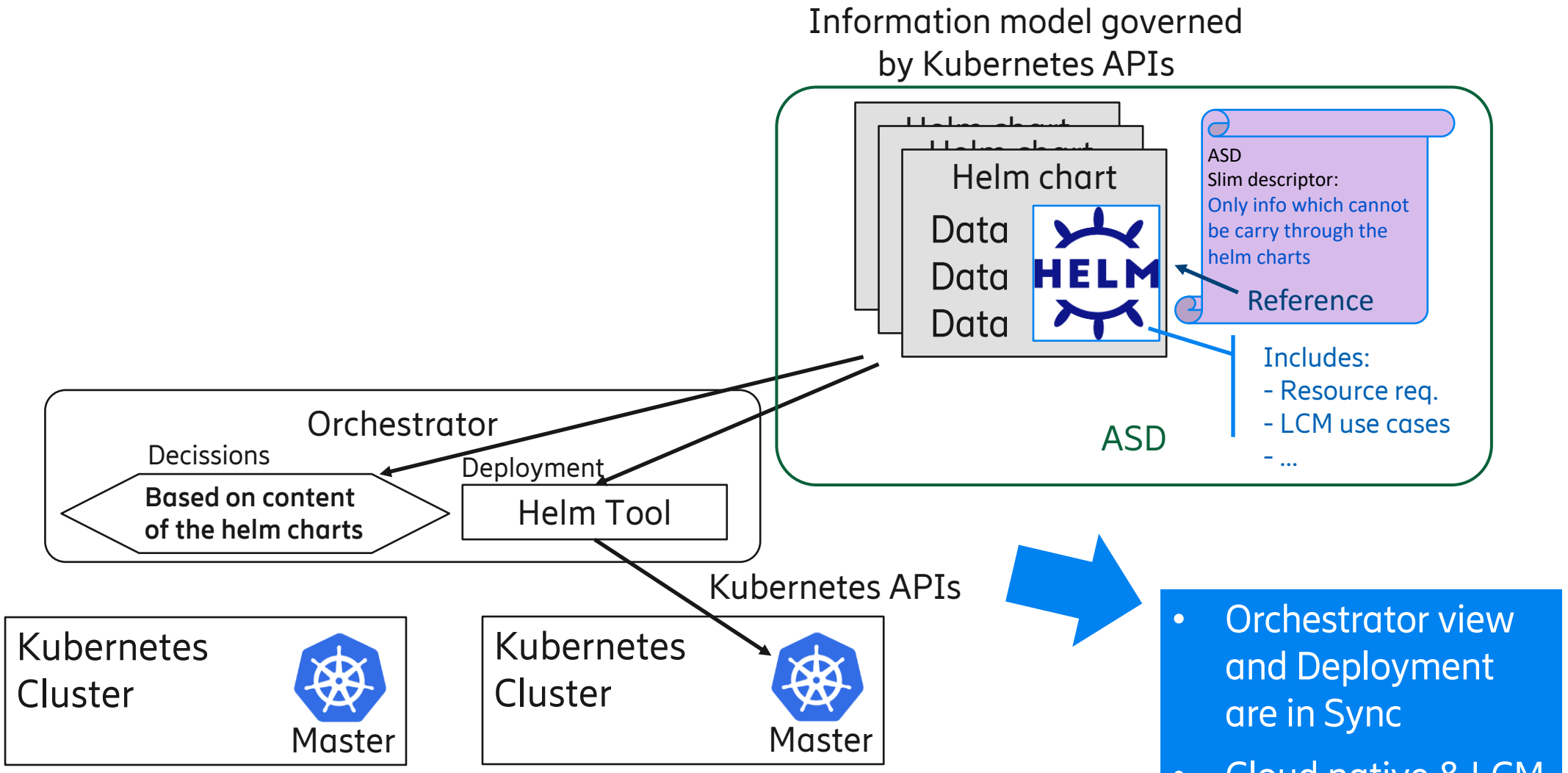
Application Service Descriptor

ASD

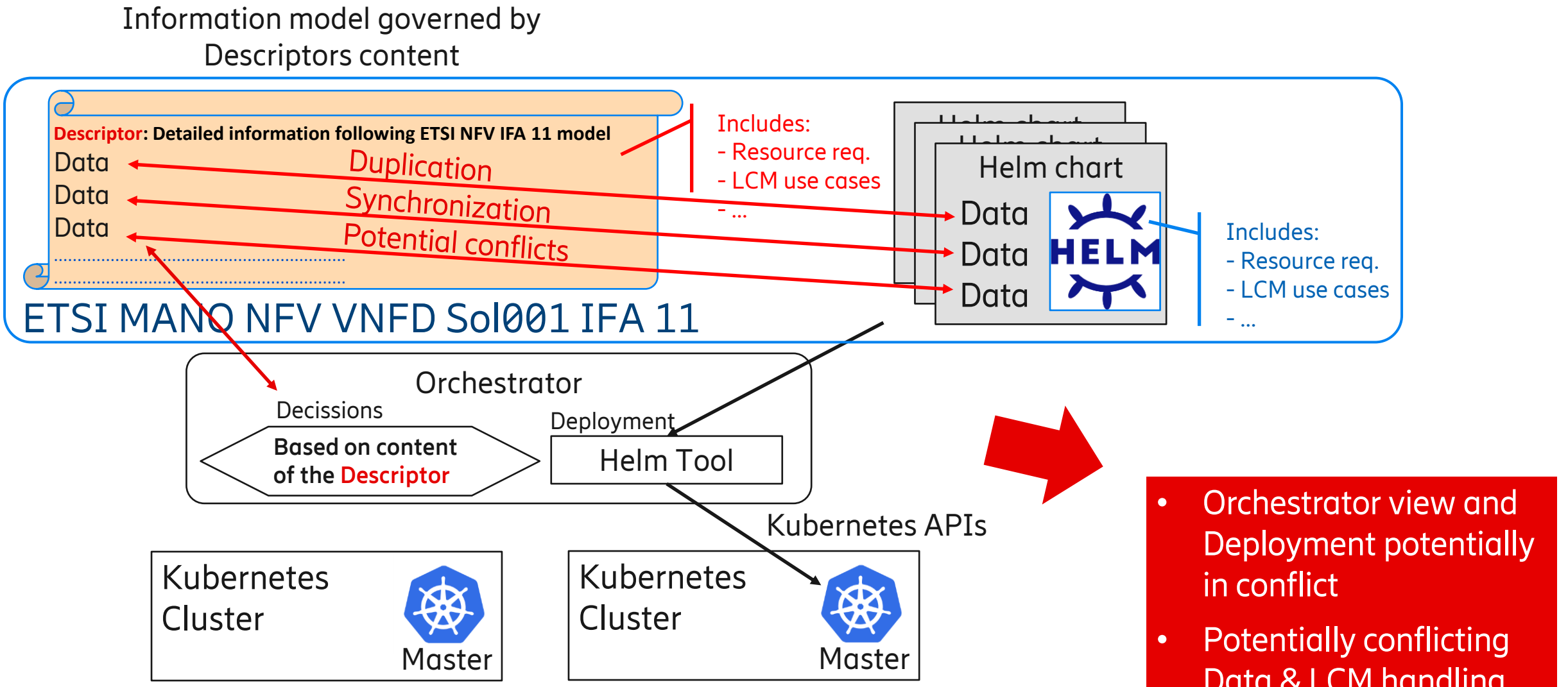
An alternative proposed by the CNF taskforce in ONAP

- Since then, the infrastructure has become more capable and taken significant SW LCM functionality and it is captured in the Helm chart.
- The VNFD is not required.
- ASD builds upon cloud native approach and tooling.

ASD based modeling and deployment of CNFs



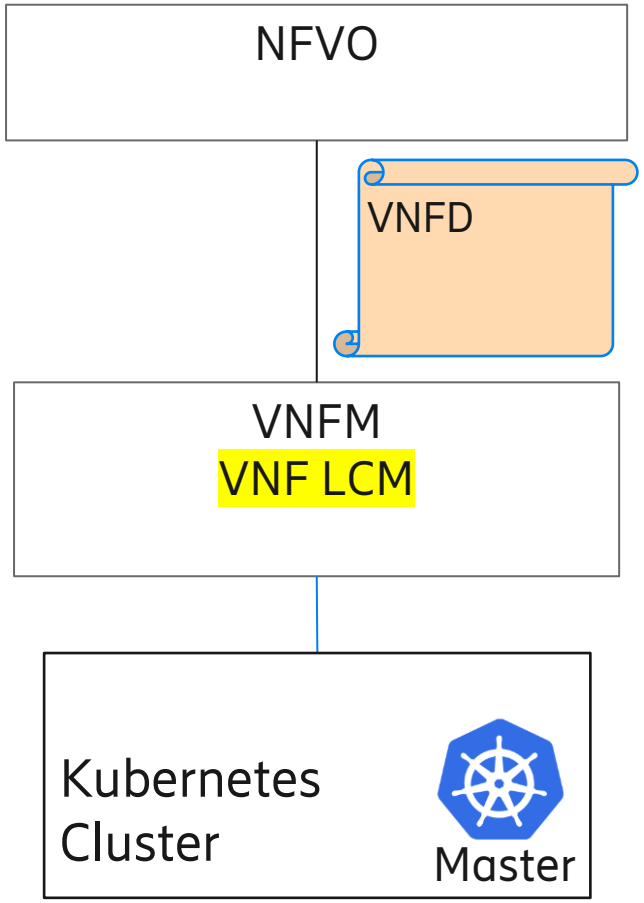
ETSI MANO NFV VNFD Sol001 IFA 11 approach



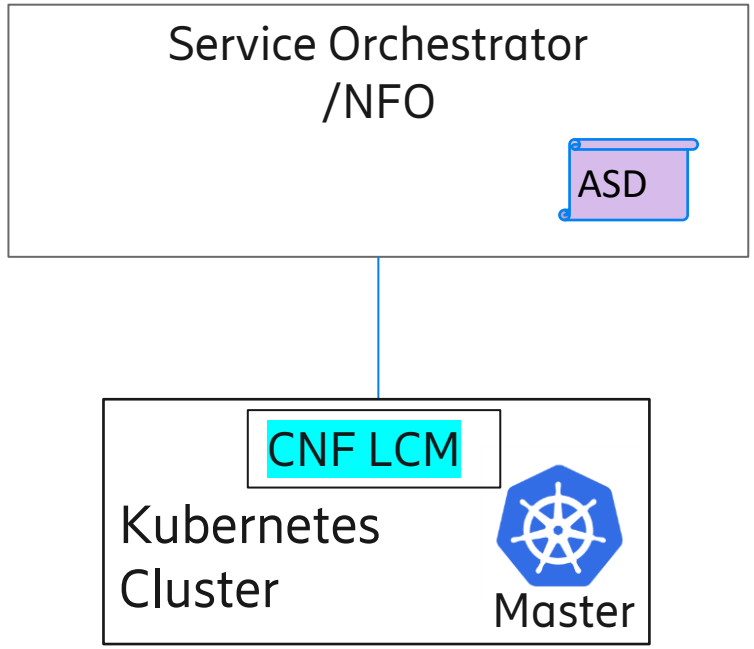
- Orchestrator view and Deployment potentially in conflict
- Potentially conflicting Data & LCM handling

Comparing LCM handling

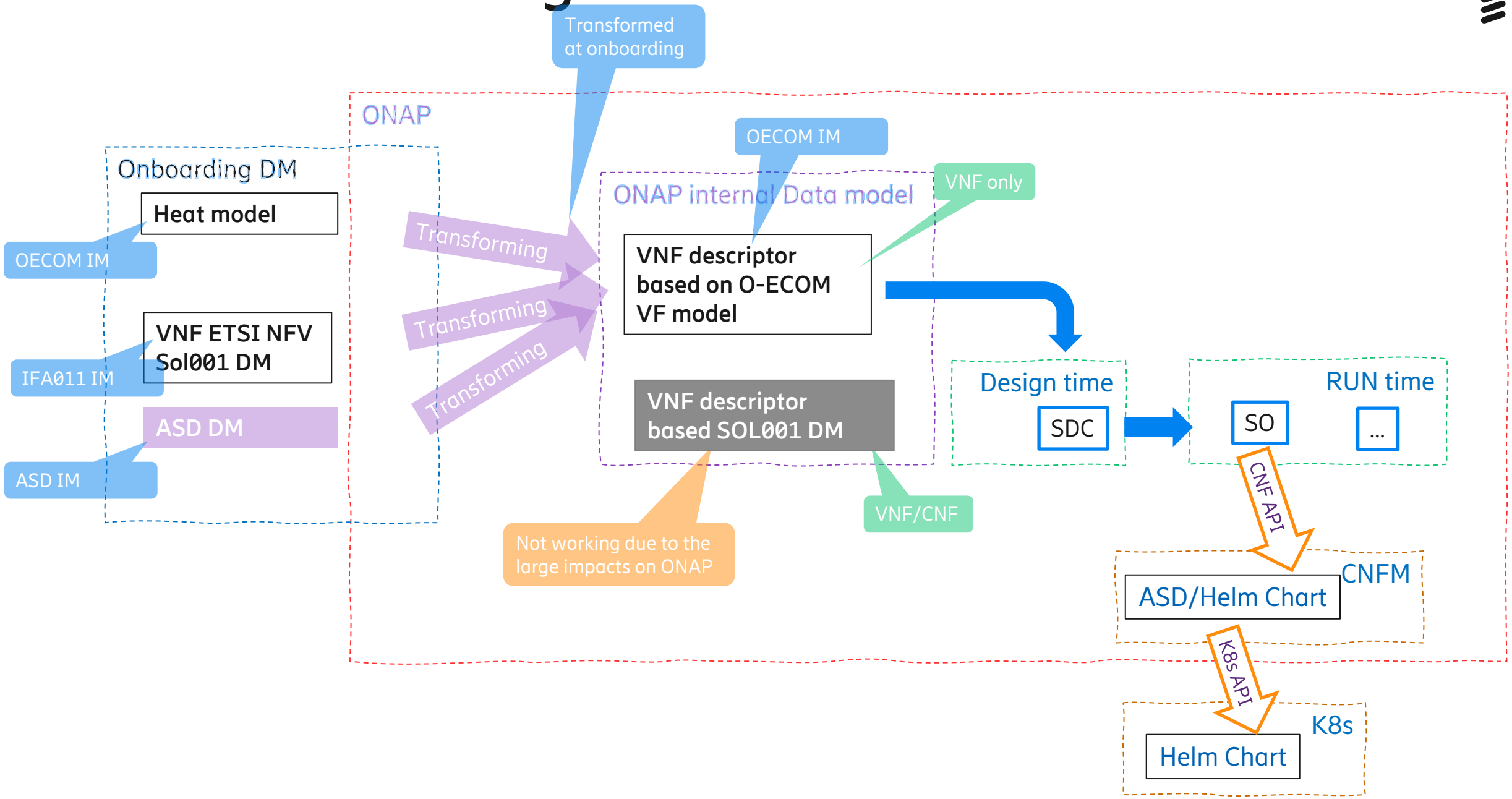
ETSI MANO NFV VNFD Sol001 IFA 11 approach



ASD based approach



ONAP xNF modelling with ASD

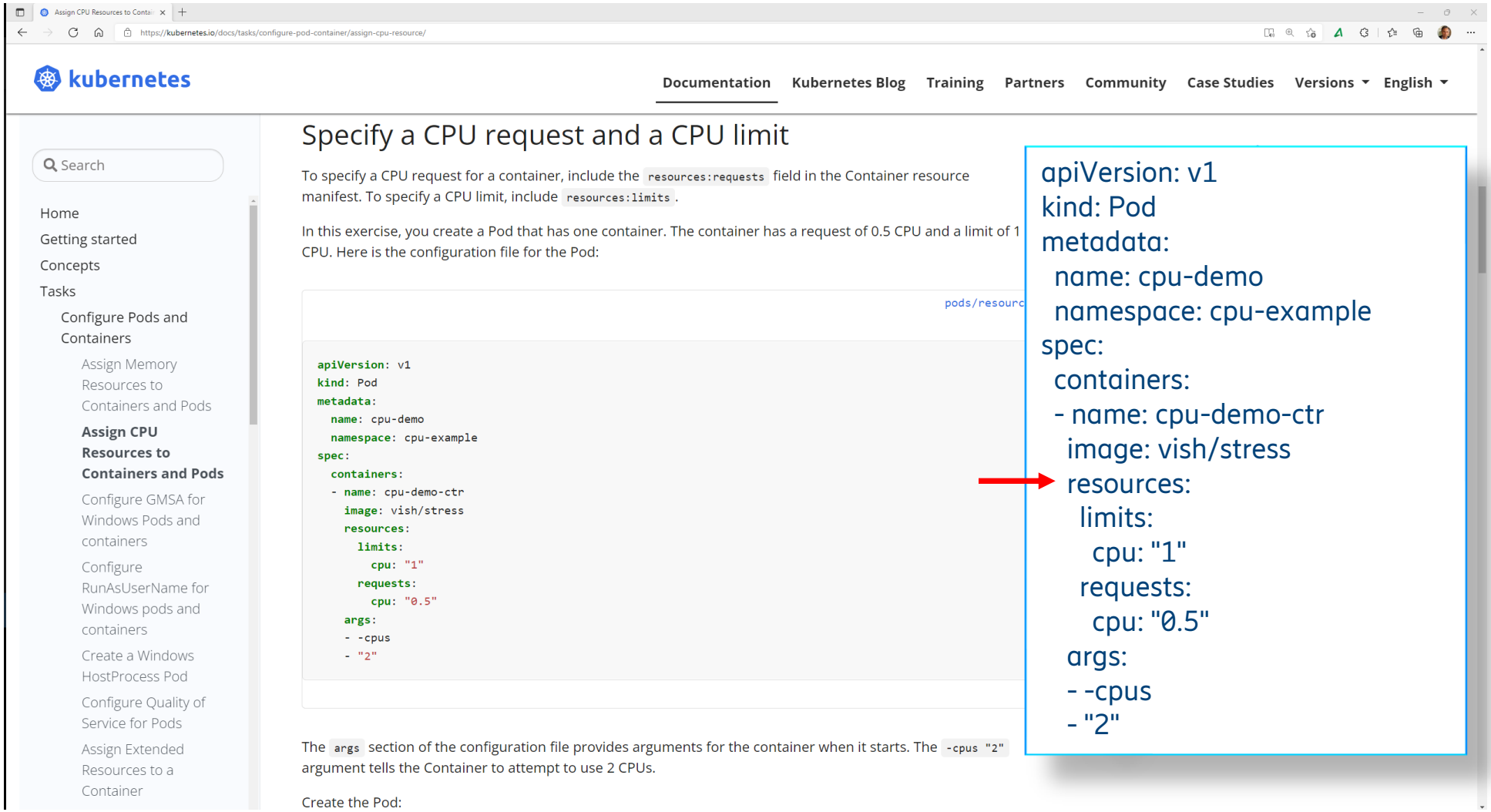


Response to comments

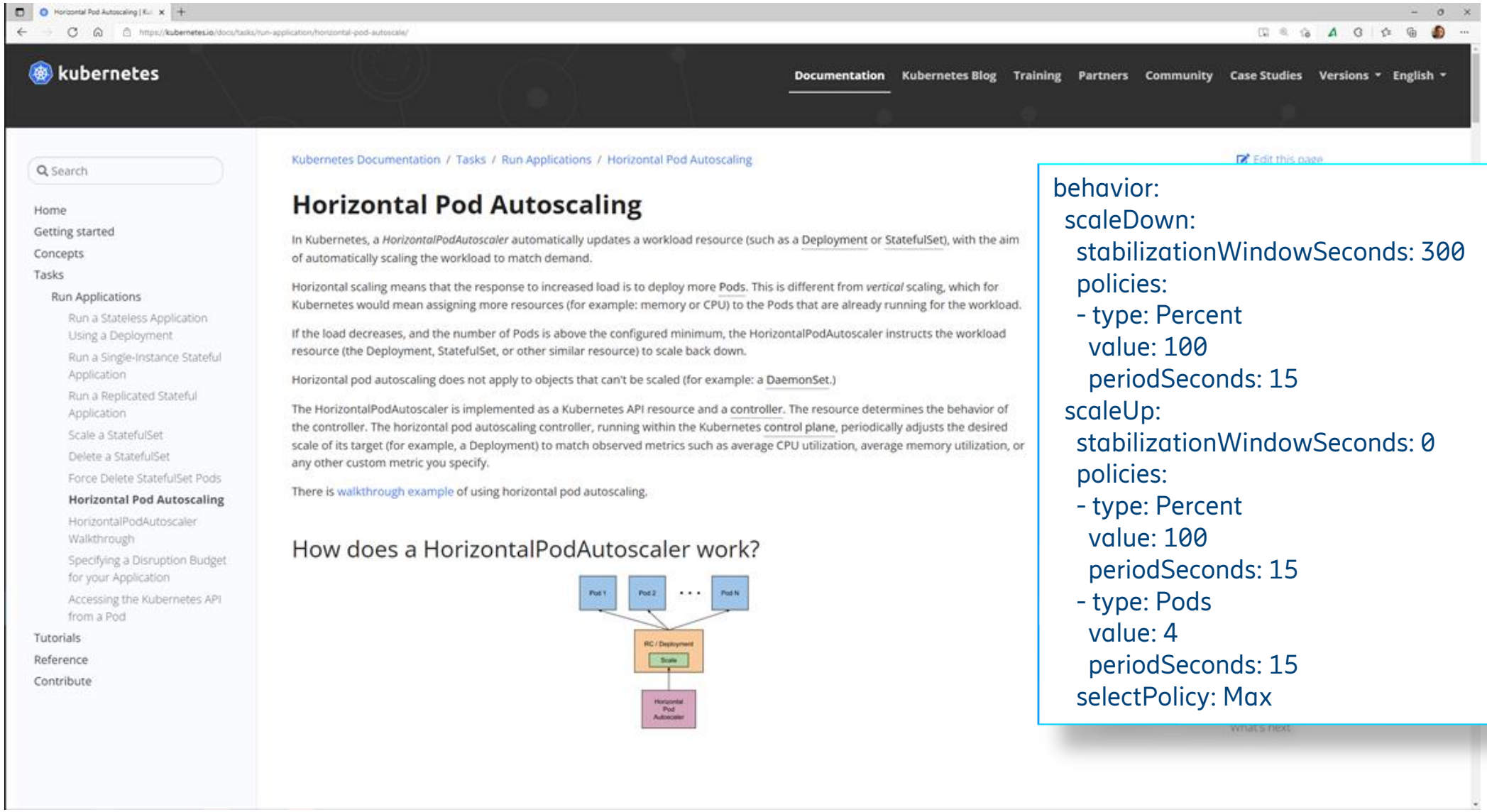
- CNF resource IM is specified by particular cloud native specifications, e.g., K8s.
- For instance, this K8s document contains all the information listed in the right side. All the info is managed by K8s, not SO.
- Proposal: adding a reference under ***Deployment Item Information Element***, that "all cloud native CNF resource IM is specified by particular cloud native specifications. K8s document is an example."

Fred's comments
Missing resource (CPU, Memory, Storage, ...) requirement information
Missing L2/L3 Protocol and Address information
Missing Software Image information
Missing Security Rules information
Missing Affinity/Anti-Affinity information
Missing Scaling Information
Missing Healing information
Missing Service/VIP mapping information
Missing configuration information
Missing monitoring information
Missing upgrade/downgrade information

Kubernetes: Resource requirement (CPU, Memory, Storage, ...) requirement information *LINK*

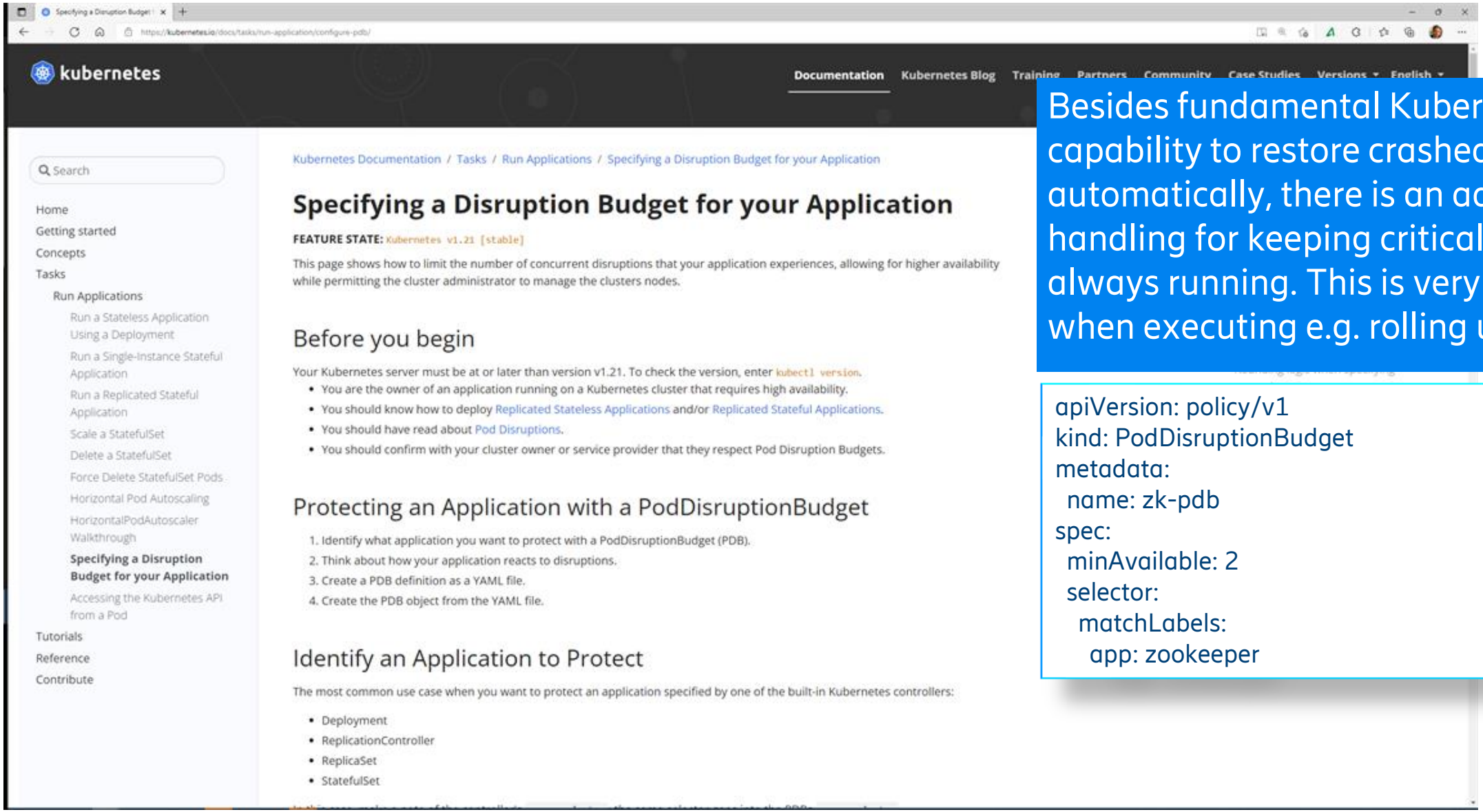


Kubernetes: Scaling Information *LINK*




```
behavior:
  scaleDown:
    stabilizationWindowSeconds: 300
    policies:
    - type: Percent
      value: 100
      periodSeconds: 15
  scaleUp:
    stabilizationWindowSeconds: 0
    policies:
    - type: Percent
      value: 100
      periodSeconds: 15
    - type: Pods
      value: 4
      periodSeconds: 15
    selectPolicy: Max
```

Kubernetes: Healing information *LINK*



Besides fundamental Kubernetes capability to restore crashed Pods automatically, there is an advanced handling for keeping critical replicas always running. This is very useful when executing e.g. rolling upgrade.

```
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: zk-pdb
spec:
  minAvailable: 2
  selector:
    matchLabels:
      app: zookeeper
```

Q&A

Did we answer your all questions?