RAFAY

Best Practices Checklist for Getting Started with Kubernetes

This checklist is designed to assist IT and DevOps professionals in creating an enterprise-grade Kubernetes management practice all the way from code to cloud. The checklist offers guidance in the form of critical requirements and best practices across four key areas: code registries, CI/CD integration, Kubernetes clusters lifecycle management and workload administration. Workload administration also covers areas of DevOps effort such as secrets management, monitoring and log aggregation. These best practices were compiled from dozens of enterprise Kubernetes deployments by Rafay's team of certified Kubernetes administrators and solution architects.

1 - Git/Source Code

		REQUIREMENT	BEST PRACTICE
1.1	**Logging in Code**	Ensure that developers write code so that "sensitive" data such as secrets are not logged/emitted to stdout/stderr.	Implement internal training and awareness programs to ensure that they implement this as a best practice.
1.2	**Secrets in YAML/Helm**	Ensure that application secrets are not embedded in your Helm charts and/or K8s YAML files.	Implement internal training and awareness programs to ensure that they implement this as a best practice.

2 - CI System

		REQUIREMENT	BEST PRACTICE
2.1	**Scanning of "Your" Container Images**	Ensure that all your container images are scanned for vulnerabilities before they are uploaded to your repo (container registry).	Embed a container image scanning tool such as Clair or Anchore into your pipeline.
2.2	**Container Base OS**	Ensure that you use a lightweight base OS for your container image. Ensure that you embed required shell, tools for interactive debugging etc if/when required.	Best Practice by Customer.
2.3	**Scanning of "3rd Party" Container Images**	Ensure that you have a review/approval process for 3rd party container images. For example, replicate these to your container registry and scan them before using them in your clusters.	Embed a container image scanning tool such as Clair or Anchore into your pipeline.

3 - Kubernetes Cluster Level

		REQUIREMENT	BEST PRACTICE
3.1	**HA for K8s Master**	Ensure that the K8s Master is architected and deployed in a "multi master" HA configuration.	Best of breed Managed K8s providers such as Amazon EKS deploy the K8s master in a HA configuration with the three masters in different AZs in the region. For users interested in upstream K8s, a best of breed Kubernetes management solution such as Rafay deploys upstream K8s in a multi master, HA configuration by default. K8s upgrades can be performed without any loss in control plane capabilities.
3.2	**etcd Security**	Ensure that only the kube API server has access to etcd service.	Best of breed Managed K8s providers such as Amazon EKS implement best practices. For users interested in upstream K8s, a best of breed Kubernetes management solution such as Rafay deploys etcd in a "secure by default" configuration so that only the kube API server can access it over a mutually authenticated TLS connection.
3.3	**etcd Encryption**	Ensure that the etcd database is encrypted and secure.	Best of breed Managed K8s providers such as Amazon EKS implement best practices. For users interested in upstream K8s, a best of breed Kubernetes management solution such as Rafay ensures that data stored in etcd is encrypted using strong encryption.
3.4	**etcd Replication**	Ensure that etcd data is replicated over a secure channel.	Best of breed Managed K8s providers such as Amazon EKS implement best practices. For users interested in upstream K8s, a best of breed Kubernetes management solution such as Rafay ensures that etcd replication is always performed over a mutually authenticated TLS channel.
3.5	**K8s Version**	Ensure that you always provision a cluster with the latest supported version of K8s.	For users interested in upstream K8s, a best of breed Kubernetes management solution such as Rafay provides this as a turn key, managed service for customers ensuring that they are always running a supported version of K8s.
3.6	**K8s Upgrades**	Ensure that your cluster has the latest patches deployed. If possible, ensure you are able to upgrade to the latest version of K8s.	Use a best of breed Kubernetes management solution such as Rafay that provides this as a built in capability/service.
3.7	**Node OS Upgrades**	Ensure that the master and worker node OS are kept up to date.	Use a best of breed Kubernetes management solution such as Rafay which provides turnkey workflows to automate this for both "Managed K8s services such as Amazon EKS" as well as "Upstream K8s" deployments.
3.8	**Visibility & Monitoring**	Ensure that cluster administrators have complete visibility and insight into all aspects of the fleet of K8s clusters.	Use a best of breed Kubernetes management solution such as Rafay or a speciality monitoring tool to get deep visibility and monitoring across the entire fleet of Kubernetes clusters.
3.9	**Capacity Monitoring**	Ensure that critical cluster metrics are monitored and tracked over time.	Use a best of breed Kubernetes management solution such as Rafay or a speciality monitoring tool to get deep visibility and monitoring across the entire fleet of Kubernetes clusters. Ensure that you have access and visibility into long term metrics required for capacity monitoring and forecasting.
3.10	**Capacity Management**	Ensure that there is a proactive capacity management strategy for the Kubernetes cluster where additional worker nodes are spun up ahead of demand. Users should understand that Kubernetes Cluster Auto Scaler is not a viable scenario for all scenarios because it will add worker nodes ONLY based on pod pending events which may not be suitable for all types of applications. The cluster auto scaler will also spin down unnecessary nodes which will evict pods on impacted nodes. Kubernetes will restart the pods on functioning nodes. But, ensure your workloads are designed to handle this type of downtime.	Use a best of breed Kubernetes management solution such as Rafay or a speciality monitoring tool to get deep visibility and monitoring across the entire fleet of Kubernetes clusters. Use the Kubernetes management solution's workflows to manage cluster capacity (add/remove workers).

		REQUIREMENT	BEST PRACTICE
3.11	**Secure Access to K8s API Server**	Ensure that the kube API server is not visible on the Internet or the Intranet to ensure that random users cannot probe it.	Use a best of breed Kubernetes management solution such as Rafay to implement a Zero Trust Access model for accessing the Kubernetes API server based on a robust RBAC strategy.
3.12	**Critical Namespaces**	Ensure that the "kube-system" and other "critical" namespaces are not available for non-privileged users and for workload deployments.	Use a best of breed Kubernetes management solution such as Rafay to implement and manage RBAC for user access to clusters.
3.13	**Multiple Namespaces**	Ensure that different workloads/ teams are provided with "separate" namespaces. Ensure no workloads are deployed in the "default" namespace.	Use a best of breed Kubernetes management solution such as Rafay to implement and manage RBAC for user access to clusters.
3.14	**Namespace Quotas**	Ensure that namespaces are configured with "quotas". This will ensure that one namespace cannot take over a cluster.	Use a best of breed Kubernetes management solution such as Rafay to implement and manage namespace quotas on your fleet of clusters.
3.15	**Cluster Audits**	Ensure that K8s cluster level auditing is enabled.	Use a best of breed Kubernetes management solution such as Rafay to enforce cluster level audits and provide visibility into all the activities on the cluster.
3.16	**RBAC**	Ensure that ONLY privileged admins have cluster admin role. If other users require kubectl access, ensure that their access is limited to a specific namespace they need access to. Enable cluster audits.	Use a best of breed Kubernetes management solution such as Rafay to implement and manage RBAC for user access to clusters. Ensure that the Kubernetes management solution provides a zero trust access model where inbound access to the K8s API server is not required.
3.17	**Dangling Resources/ Objects**	Ensure you have a proactive strategy to clean up "dangling" resources or objects in your clusters. For example, developers can forget about PVCs.	Use a best of breed Kubernetes management solution such as Rafay to get visibility into dangling resources/objects and clean them up on a periodic basis.
3.18	**Pod Security Policy (PSP)**	Ensure that you have appropriate PSPs enabled on the cluster to prevent malicious pods from operating.	Use a best of breed Kubernetes management solution such as Rafay to manage and enforce Kubernetes PSPs on your fleet of clusters.
3.19	**Network Policies**	Ensure that you have required Network Policies configured especially to secure (a) namespace to namespace and (b) namespace to external services communications.	Customer Best Practice.
3.20	**CIS Benchmarks**	Ensure you run a 3rd party CIS Benchmarking tool such as kube-bench to detect and remediate compliance related issues with your K8s cluster.	Customer Best Practice.
3.21	**Pod Disruption Budget (PDB)**	Ensure to specify a pod disruption budget for your application so that it can tolerate pod evictions without causing outage. This will come in handy when performing upgrades on worker nodes.	Customer Best Practice.
3.22	**Node Termination Handlers**	On public cloud environments such as AWS, leverage tools such as node termination handlers to ensure that the cluster can gracefully handle unforeseen issues with the nodes/ instances.	Customer Best Practice. For Rafay provisioned Amazon EKS Clusters, Rafay automatically deploys the AWS Node Termination Handler daemonset as part of the default cluster blueprint.

4 - Workload Level

		REQUIREMENT	BEST PRACTICE
4.1	**Container Images**	Do not use “latest” for container image tags.	Customer Best Practice.
4.2	**Custom Helm/YAML**	For Helm workloads, ensure that the values.yaml exposes all required configuration settings (see list below) so that the Ops team can tune and optimize the specs independently of the the Dev team.	Customer Best Practice.
4.3	**Image Pull Policy**	Recommend using a “IfNotPresent” policy for large images. However, it is worth emphasizing that Docker is intelligent to download only updated layers. However, horizontal pod auto-scaling (HPA) will work much faster if the locally cached images are used.	Customer Best Practice.
4.4	**Requests/ Limits**	Ensure that all containers have “requests” and “limits” configured in the specs. At a minimum ensure that limits are specified.	Customer Best Practice. Ensure that you use a best of breed Kubernetes management solution such as Rafay to provide DevOps teams visibility into pods that are incorrectly configured.
4.5	**Initial Replicas**	Ensure that a sufficient baseline set of replicas are configured to operate especially if your containers are large and slow to start.	Customer Best Practice.
4.6	**HPA**	Ensure that horizontal pod autoscaling (HPA) is configured to scale the pods based on demand.	Customer Best Practice.
4.7	**Log Aggregation**	Ensure that logs are automatically aggregated from deployed containers to a centralized log management system such as ElasticSearch. Provide developers access to their container logs through the log management system’s interface. For example, Kibana for Elasticsearch.	Customer Best Practice.
4.8	**Secrets**	Ensure that secrets are not manually handled or embedded in YAML or Helm charts. Consider using a JIT provisioning/deprovisioning workflow if using K8s Secrets infrastructure. If possible, leverage a secrets management infrastructure like Vault where the pods will retrieve secrets and store ONLY in memory.	Leverage a centralized secrets management system and have your workload’s pods retrieve the secrets dynamically. Ensure that you use a best of breed Kubernetes management solution such as Rafay which provides turn key integrations with secrets management platforms such as Vault.
4.9	**Rolling Updates**	Strongly recommend using a “Rolling Update” strategy to avoid downtime of customer applications. Ensure that the cluster has “sufficient” capacity available to handle the extra capacity needed for rolling updates.	Customer Best Practice.

		REQUIREMENT	BEST PRACTICE
4.10	**Specs in YAML or Helm**	Recommend that you maintain a single, integrated YAML or Helm chart instead of 10s or 100s of separate YAML files.	Customer Best Practice.
4.11	**Readiness and Liveness Probes**	Strongly recommend the use of liveness and readiness probes. Readiness probes ensure that K8s will stop sending traffic to this pod immediately and ensure the uptime of your application. Liveness probes ensure that K8s has the means to verify whether your pod is alive or not. If this probe fails, K8s will know that it has to restart the pod.	Customer Best Practice.
4.12	**Size of ConfigMaps**	Ensure that your ConfigMaps are not larger than 1MB. ConfigMaps are stored in the etcd database in Kubernetes that imposes a size limit of 1MB.	Customer Best Practice.

CLOUD NATIVE COMPUTING FOUNDATION SILVER MEMBER
kubernetes CERTIFIED SERVICE PROVIDER
certified kubernetes
aws partner network | Select Technology Partner

© 2021 Rafay Systems, Inc. All rights reserved. All other trademarks and service marks are property of their respective owners.