(/)

Zero-downtime Deployment in Kubernetes with Jenkins

Monday, April 30, 2018

Zero-downtime Deployment in Kubernetes with Jenkins (https://kubernetes.io/blog/2018/04/30/zero-downtime-deployment-kubernetes-jenkins/)

Ever since we added the Kubernetes Continuous Deploy (https://aka.ms/azjenkinsk8s) and Azure Container Service (https://aka.ms/azjenkinsacs) plugins to the Jenkins update center, “How do I create zero-downtime deployments” is one of our most frequently-asked questions. We created a quickstart template on Azure to demonstrate what zero-downtime deployments can look like. Although our example uses Azure, the concept easily applies to all Kubernetes installations.

Rolling Update

Kubernetes supports the RollingUpdate strategy to replace old pods with new ones gradually, while continuing to serve clients without incurring downtime. To perform a RollingUpdate deployment:

- Set `.spec.strategy.type` to `RollingUpdate` (the default value).
- Set `.spec.strategy.rollingUpdate.maxUnavailable` and `.spec.strategy.rollingUpdate.maxSurge` to some reasonable value.
    - `maxUnavailable` : the maximum number of pods that can be unavailable during the update process. This can be an absolute number or percentage of the replicas count; the default is 25%.
    - `maxSurge` : the maximum number of pods that can be created over the desired number of pods. Again this can be an absolute number or a percentage of the replicas count; the default is 25%.
- Configure the `readinessProbe` for your service container to help Kubernetes determine the state of the pods. Kubernetes will only route the client traffic to the pods with a healthy liveness probe.

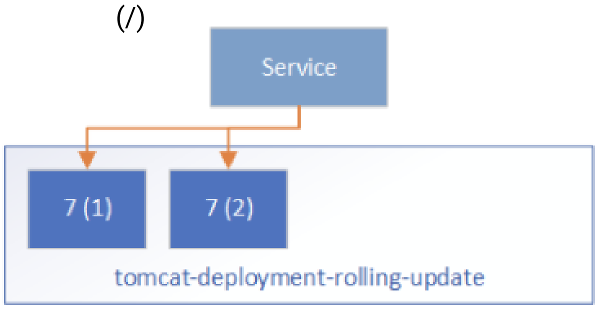
We’ll use deployment of the official Tomcat image to demonstrate this:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment-rolling-update
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: tomcat
        role: rolling-update
    spec:
      containers:
      - name: tomcat-container
        image: tomcat:${TOMCAT_VERSION}
        ports:
        - containerPort: 8080
        readinessProbe:
          httpGet:
            path: /
            port: 8080
  strategy:
    type: RollingUpdate
    rollingUp      maxSurge: 50%
```

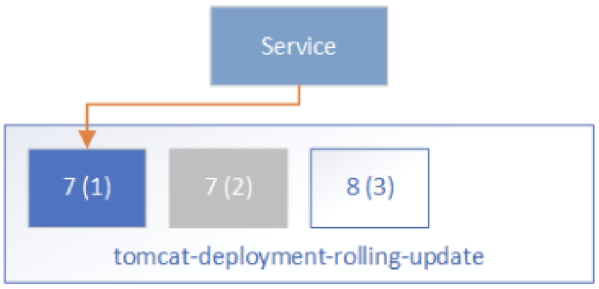
If the Tomcat running in the current deployments is version 7, we can replace `${TOMCAT_VERSION}` with 8 and apply this to the Kubernetes cluster. With the Kubernetes Continuous Deploy (https://aka.ms/azjenkinsk8s) or the Azure Container Service (https://aka.ms/azjenkinsacs) plugin, the value can be fetched from an environment variable which eases the deployment process.

Behind the scenes, Kubernetes manages the update like so:

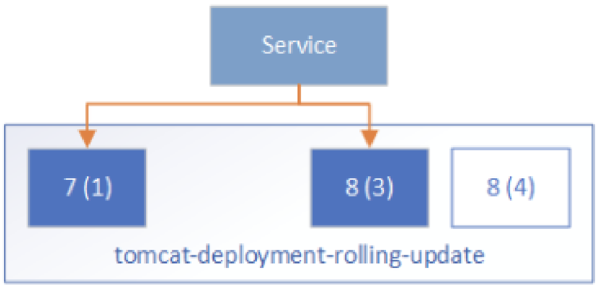
(/)



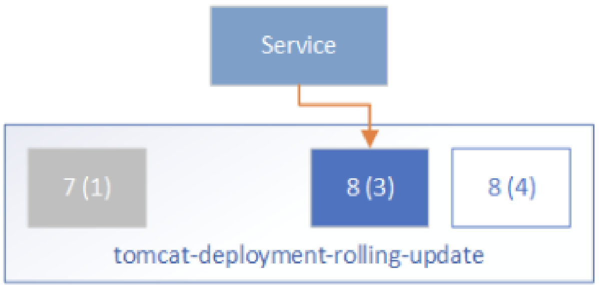
1. initial state



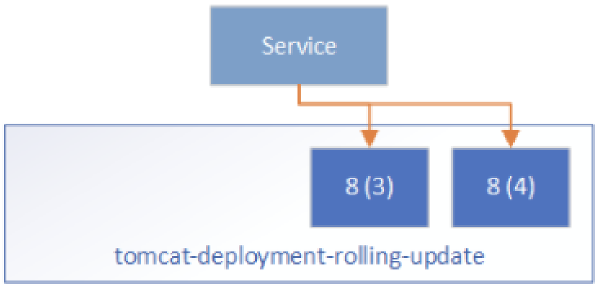
2. turn off old pods and create new ones



3. serving both versions



4. further update



5. update complete

- Initially, all pods are running Tomcat 7 and the frontend Service routes the traffic to these pods.
- During the rolling update, Kubernetes takes down some Tomcat 7 pods and creates corresponding new Tomcat 8 pods. It ensures:
    - at most `maxUnavailable` pods in the desired Pods can be unavailable, that is, at least (`replicas - maxUnavailable`) pods should be serving the client traffic, which is 2-1=1 in our case.
    - at most maxSurge more pods can be created during the update process, that is 2*50%=1 in our case.
- One Tomcat 7 pod is taken down, and one Tomcat 8 pod is created. Kubernetes will not route the traffic to any of them because their readiness probe is not yet successful.
- When the new Tomcat 8 pod is ready as determined by the readiness probe, Kubernetes will start routing the traffic to it. This means during the update process, users may see both the old service and the new service.
- The rolling update continues by taking down Tomcat 7 pods and creating Tomcat 8 pods, and then routing the traffic to the ready pods.
- Finally, all pods are on Tomcat 8.

The Rolling Update strategy ensures we always have some Ready backend pods serving client requests, so there's no service downtime. However, some extra care is required: - During the update, both the old pods and new pods may serve the requests. Without well defined session affinity in the Service layer, a user may be routed to the new pods and later back to the old pods. - This also requires you to maintain well-defined forward and backward compatibility for both data and the API, which can be challenging. - It may take a long time before a pod is ready for traffic after it is started. There may be a long window of time where the traffic is served with less backend pods than usual. Generally, this should not be a problem as we tend to do production upgrades when the service is less busy. But this will also extend the time window for issue 1. - We cannot do comprehensive tests for the new pods being created. Moving application changes from dev / QA environments to production can represent a persistent risk of breaking existing functionality. The readiness probe can do some work to check readiness, however, it should be a lightweight task that can be run periodically, and not suitable to be used as an entry point to start the complete tests.

Blue/green Deployment

Blue/green deployment quoted from TechTarget

> A blue/green deployment is a change management strategy for releasing software code. Blue/green deployments, which may also be referred to as A/B deployments require two identical hardware environments that are configured exactly the same way. While one environment is active and serving end users, the other environment remains idle.

Container technology offers a stand-alone environment to run the desired service, which makes it super easy to create identical environments as required in the blue/green deployment. The loosely coupled Services - ReplicaSets, and the label/selector-based service routing in Kubernetes make it easy to switch between different backend environments. With these techniques, the blue/green deployments in Kubernetes can be done as follows:

- Before the deployment, the infrastructure is prepared like so:
    - Prepare the blue deployment and green deployment with `TOMCAT_VERSION=7` and `TARGET_ROLE` set to blue or green respectively.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment-${TARGET_ROLE}
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: tomcat
        role: ${TARGET_ROLE}
    spec:
      containers:
      - name: tomcat-container
        image: tomcat:${TOMCAT_VERSION}
        ports:
        - containerPort: 8080
        readinessProbe:
          httpGet:
            path: /
            port: 8080
```

- Prepare the public service endpoint, which initially routes to one of the backend environments, say `TARGET_ROLE=blue`.

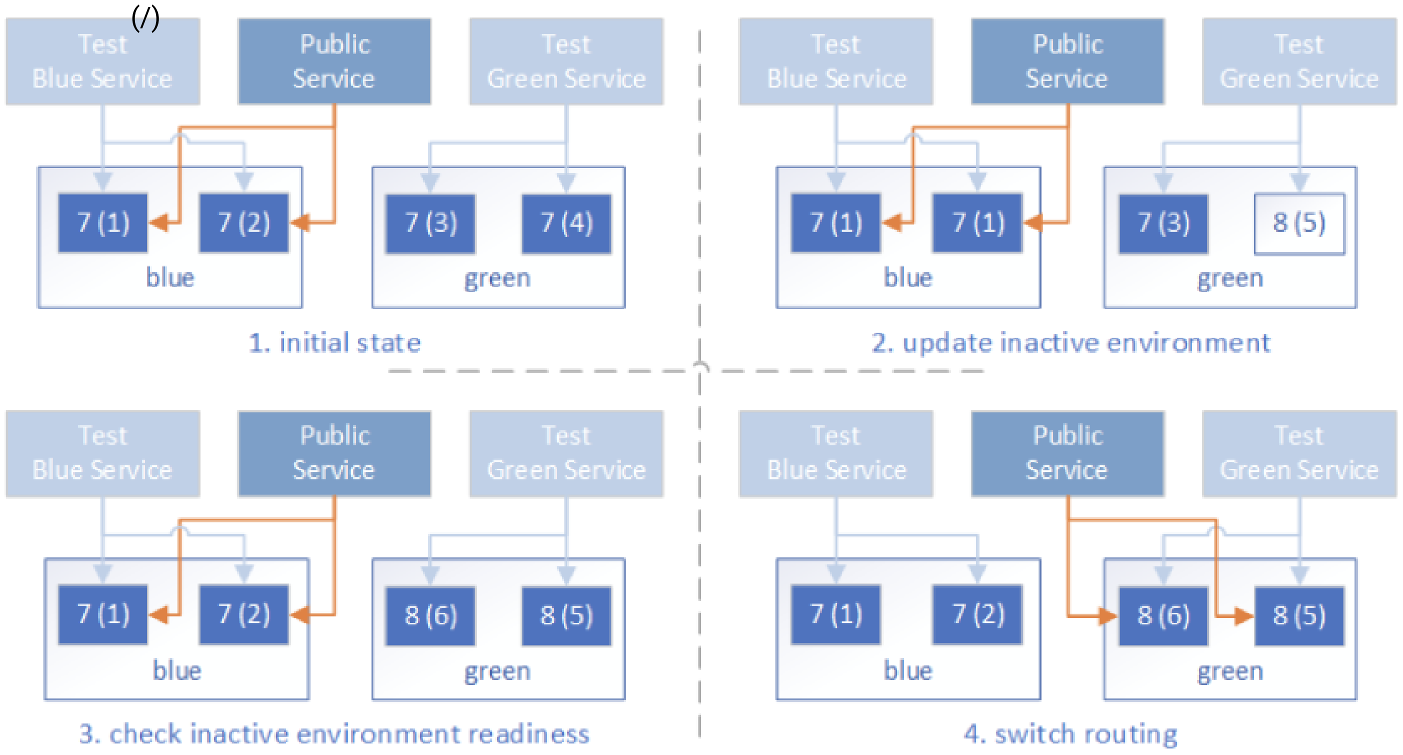
```
kind: Service
apiVersion: v1
metadata:
  name: tomcat-service
  labels:
    app: tomcat
    role: ${TARGET_ROLE}
    env: prod
spec:
  type: LoadBalancer
  selector:
    app: tomcat
    role: ${TARGET_ROLE}
  ports:
    - port: 80
      targetPort: 8080
```

- Optionally, prepare a test endpoint so that we can visit the backend environments for testing. They are similar to the public service endpoint, but they are intended to be accessed internally by the dev/ops team only.

```
kind: Service
apiVersion: v1
metadata:
  name: tomcat-test-${TARGET_ROLE}
  labels:
    app: tomcat
    role: test-${TARGET_ROLE}
spec:
  type: LoadBalancer
  selector:
    app: tomcat
    role: ${TARGET_ROLE}
  ports:
    - port: 80
      targetPort: 8080
```

- Update the application in the inactive environment, say green environment. Set `TARGET_ROLE=green` and `TOMCAT_VERSION=8` in the deployment config to update the green environment.
- Test the deployment via the `tomcat-test-green` test endpoint to ensure the green environment is ready to serve client traffic.
- Switch the frontend Service routing to the green environment by updating the Service config with `TARGET_ROLE=green` .
- Run additional tests on the public endpoint to ensure it is working properly.
- Now the blue environment is idle and we can:
    - leave it with the old application so that we can roll back if there's issue with the new application
    - update it to make it a hot backup of the active environment
    - reduce its replica count to save the occupied resources

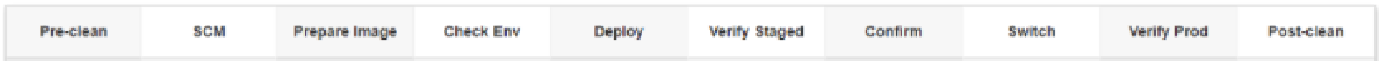
(/)



As compared to Rolling Update, the blue/green up* The public service is either routed to the old applications, or new applications, but never both at the same time. * The time it takes for the new pods to be ready does not affect the public service quality, as the traffic will only be routed to the new pods when all of them are tested to be ready. * We can do comprehensive tests on the new environment before it serves any public traffic. Just keep in mind this is in production, and the tests should not pollute live application data.

Jenkins Automation

Jenkins provides easy-to-setup workflow to automate your deployments. With Pipeline (https://jenkins.io/doc/book/pipeline/) support, it is flexible to build the zero-downtime deployment workflow, and visualize the deployment steps. To facilitate the deployment process for Kubernetes resources, we published the Kubernetes Continuous Deploy (https://aka.ms/azjenkinsk8s) and the Azure Container Service (https://aka.ms/azjenkinsacs) plugins built based on the kubernetes-client (https://github.com/fabric8io/kubernetes-client). You can deploy the resource to Azure Kubernetes Service (AKS) or the general Kubernetes clusters without the need of kubectl, and it supports variable substitution in the resource configuration so you can deploy environment-specific resources to the clusters without updating the resource config. We created a Jenkins Pipeline to demonstrate the blue/green deployment to AKS. The flow is like the following:



- Pre-clean: clean workspace.
- SCM: pulling code from the source control management system.
- Prepare Image: prepare the application docker images and upload them to some Docker repository.
- Check Env: determine the active and inactive environment, which drives the following deployment.
- Deploy: deploy the new application resource configuration to the inactive environment. With the Azure Container Service plugin, this can be done with: `acsDeploy azureCredentialsId: 'stored-azure-credentials-id', configFilePaths: "glob/path/to/*/resource-config-*.yml", containerService: "aks-name | AKS", resourceGroupName: "resource-group-name", enableConfigSubstitution: true`
- Verify Staged: verify the deployment to the inactive environment to ensure it is working properly. Again, note this is in the production environment, so be careful not to pollute live application data during tests.
- Confirm: Optionally, send email notifications for manual user approval to proceed with the actual environment switch.

- Switch: Switch the frontend service endpoint routing to the inactive environment. This is just another service deployment to the AKS Kubernetes cluster.
- Verify Prod: verify the frontend service endpoint is working properly with the new environment.
- Post-clean: do some post clean on the temporary files.

For the Rolling Update strategy, simply deploy the deployment configuration to the Kubernetes cluster, which is a simple, single step.

Put It All Together

We built a quickstart template on Azure to demonstrate how we can do the zero-downtime deployment to AKS (Kubernetes) with Jenkins. Go to Jenkins Blue-Green Deployment on Kubernetes (https://aka.ms/azjenkinsk8sqs) and click the button Deploy to Azure to get the working demo. This template will provision:

- An AKS cluster, with the following resources:
  - Two similar deployments representing the environments “blue” and “green”. Both are initially set up with the `tomcat` :7 image.
  - Two test endpoint services (`tomcat-test-blue` and `tomcat-test-green`), which are connected to the corresponding deployments, and can be used to test if the deployments are ready for production use.
  - A production service endpoint (`tomcat-service`) which represents the public endpoint that the users will access. Initially it is routing to the “blue” environment.
- A Jenkins master running on an Ubuntu 16.04 VM, with the Azure service principal credentials configured. The Jenkins instance has two sample jobs:
  - AKS Kubernetes Rolling Update Deployment pipeline to demonstrate the Rolling Update deployment to AKS.
  - AKS Kubernetes Blue/green Deployment pipeline to demonstrate the blue/green deployment to AKS.
  - We didn’t include the email confirmation step in the quickstart template. To add that, you need to configure the email SMTP server details in the Jenkins system configuration, and then add a Pipeline stage before Switch: `stage('Confirm') { mail (to: 'to@example.com', subject: "Job '${env.JOB_NAME}' (${env.BUILD_NUMBER}) is waiting for input", body: "Please go to ${env.BUILD_URL}.") input 'Ready to go?' }` Follow the Steps (https://github.com/Azure/azure-quickstart-templates/tree/master/301-jenkins-aks-zero-downtime-deployment#steps) to setup the resources and you can try it out by start the Jenkins build jobs.

« Prev (/blog/2018/04/25/open-source-charts-2017/)

Next >> (/blog/2018/05/01/developing-on-kubernetes/)

(https://kubernetes.io/feed.xml)

@Kubernetesio (https://twitter.com/kubernetesio)

View on Github (https://github.com/kubernetes/kubernetes)

(/)
#kubernetes-users (http://slack.k8s.io)

Stack Overflow (https://stackoverflow.com/questions/tagged/kubernetes)

Forum (https://discuss.kubernetes.io)

Download Kubernetes (http://get.k8s.io/)

2018

- Tips for Your First Kubecon Presentation - Part 2 (https://kubernetes.io/blog/2018/10/26/tips-for-your-first-kubecon-presentation---part-2/) Oct 26
- Tips for Your First Kubecon Presentation - Part 1 (https://kubernetes.io/blog/2018/10/18/tips-for-your-first-kubecon-presentation---part-1/) Oct 18
- Kubernetes 2018 North American Contributor Summit (https://kubernetes.io/blog/2018/10/16/kubernetes-2018-north-american-contributor-summit/) Oct 16
- 2018 Steering Committee Election Results (https://kubernetes.io/blog/2018/10/15/2018-steering-committee-election-results/) Oct 15
- Topology-Aware Volume Provisioning in Kubernetes (https://kubernetes.io/blog/2018/10/11/topology-aware-volume-provisioning-in-kubernetes/) Oct 11
- Kubernetes v1.12: Introducing RuntimeClass (https://kubernetes.io/blog/2018/10/10/kubernetes-v1.12-introducing-runtimeclass/) Oct 10
- Introducing Volume Snapshot Alpha for Kubernetes (https://kubernetes.io/blog/2018/10/09/introducing-volume-snapshot-alpha-for-kubernetes/) Oct 9
- Support for Azure VMSS, Cluster-Autoscaler and User Assigned Identity (https://kubernetes.io/blog/2018/10/08/support-for-azure-vmss--cluster-autoscaler-and-user-assigned-identity/) Oct 8
- Introducing the Non-Code Contributor’s Guide (https://kubernetes.io/blog/2018/10/04/introducing-the-non-code-contributors-guide/) Oct 4
- KubeDirector: The easy way to run complex stateful applications on Kubernetes (https://kubernetes.io/blog/2018/10/03/kubedirector-the-easy-way-to-run-complex-stateful-applications-on-kubernetes/) Oct 3
- Building a Network Bootable Server Farm for Kubernetes with LTSP (https://kubernetes.io/blog/2018/10/02/building-a-network-bootable-server-farm-for-kubernetes-with-ltsp/) Oct 2
- Health checking gRPC servers on Kubernetes (https://kubernetes.io/blog/2018/10/01/health-checking-grpc-servers-on-kubernetes/) Oct 1

(/)

Kubernetes 1.12: Kubelet TLS Bootstrap and Azure Virtual Machine Scale Sets (VMSS) Move to General Availability (https://kubernetes.io/blog/2018/09/27/kubernetes-1.12-kubelet-tls-bootstrap-and-azure-virtual-machine-scale-sets-vmss-move-to-general-availability/) Sep 27

Hands On With Linkerd 2.0 (https://kubernetes.io/blog/2018/09/18/hands-on-with-linkerd-2.0/) Sep 18

2018 Steering Committee Election Cycle Kicks Off (https://kubernetes.io/blog/2018/09/06/2018-steering-committee-election-cycle-kicks-off/) Sep 6

The Machines Can Do the Work, a Story of Kubernetes Testing, CI, and Automating the Contributor Experience (https://kubernetes.io/blog/2018/08/29/the-machines-can-do-the-work-a-story-of-kubernetes-testing-ci-and-automating-the-contributor-experience/) Aug 29

Introducing Kubebuilder: an SDK for building Kubernetes APIs using CRDs (https://kubernetes.io/blog/2018/08/10/introducing-kubebuilder-an-sdk-for-building-kubernetes-apis-using-crds/) Aug 10

Out of the Clouds onto the Ground: How to Make Kubernetes Production Grade Anywhere (https://kubernetes.io/blog/2018/08/03/out-of-the-clouds-onto-the-ground-how-to-make-kubernetes-production-grade-anywhere/) Aug 3

Dynamically Expand Volume with CSI and Kubernetes (https://kubernetes.io/blog/2018/08/02/dynamically-expand-volume-with-csi-and-kubernetes/) Aug 2

KubeVirt: Extending Kubernetes with CRDs for Virtualized Workloads (https://kubernetes.io/blog/2018/07/27/kubevirt-extending-kubernetes-with-crds-for-virtualized-workloads/) Jul 27

Feature Highlight: CPU Manager (https://kubernetes.io/blog/2018/07/24/feature-highlight-cpu-manager/) Jul 24

The History of Kubernetes & the Community Behind It (https://kubernetes.io/blog/2018/07/20/the-history-of-kubernetes--the-community-behind-it/) Jul 20

Kubernetes Wins the 2018 OSCON Most Impact Award (https://kubernetes.io/blog/2018/07/19/kubernetes-wins-2018-oscon-most-impact-award/) Jul 19

11 Ways (Not) to Get Hacked (https://kubernetes.io/blog/2018/07/18/11-ways-not-to-get-hacked/) Jul 18

How the sausage is made: the Kubernetes 1.11 release interview, from the Kubernetes Podcast (https://kubernetes.io/blog/2018/07/16/how-the-sausage-is-made-the-kubernetes-1.11-release-interview-from-the-kubernetes-podcast/) Jul 16

Resizing Persistent Volumes using Kubernetes (https://kubernetes.io/blog/2018/07/12/resizing-persistent-volumes-using-kubernetes/) Jul 12

Dynamic Kubelet Configuration (https://kubernetes.io/blog/2018/07/11/dynamic-kubelet-configuration/) Jul 11

Meet Our Contributors - Monthly Streaming YouTube Mentoring Series (https://kubernetes.io/blog/2018/07/10/meet-our-contributors---monthly-streaming-youtube-mentoring-series/) Jul 10

CoreDNS GA for Kubernetes Cluster DNS (https://kubernetes.io/blog/2018/07/10/coredns-ga-for-kubernetes-cluster-dns/) Jul 10

(/)

IPVS-Based In-Cluster Load Balancing Deep Dive (https://kubernetes.io/blog/2018/07/09/ipvs-based-in-cluster-load-balancing-deep-dive/) Jul 9

Airflow on Kubernetes (Part 1): A Different Kind of Operator (https://kubernetes.io/blog/2018/06/28/airflow-on-kubernetes-part-1-a-different-kind-of-operator/) Jun 28

Kubernetes 1.11: In-Cluster Load Balancing and CoreDNS Plugin Graduate to General Availability (https://kubernetes.io/blog/2018/06/27/kubernetes-1.11-release-announcement/) Jun 27

Dynamic Ingress in Kubernetes (https://kubernetes.io/blog/2018/06/07/dynamic-ingress-in-kubernetes/) Jun 7

4 Years of K8s (https://kubernetes.io/blog/2018/06/06/4-years-of-k8s/) Jun 6

Say Hello to Discuss Kubernetes (https://kubernetes.io/blog/2018/05/30/say-hello-to-discuss-kubernetes/) May 30

Introducing kustomize; Template-free Configuration Customization for Kubernetes (https://kubernetes.io/blog/2018/05/29/introducing-kustomize-template-free-configuration-customization-for-kubernetes/) May 29

Kubernetes Containerd Integration Goes GA (https://kubernetes.io/blog/2018/05/24/kubernetes-containerd-integration-goes-ga/) May 24

Getting to Know Kubevirt (https://kubernetes.io/blog/2018/05/22/getting-to-know-kubevirt/) May 22

Gardener - The Kubernetes Botanist (https://kubernetes.io/blog/2018/05/17/gardener/) May 17

Docs are Migrating from Jekyll to Hugo (https://kubernetes.io/blog/2018/05/05/hugo-migration/) May 5

Announcing Kubeflow 0.1 (https://kubernetes.io/blog/2018/05/04/announcing-kubeflow-0.1/) May 4

Current State of Policy in Kubernetes (https://kubernetes.io/blog/2018/05/02/policy-in-kubernetes/) May 2

Developing on Kubernetes (https://kubernetes.io/blog/2018/05/01/developing-on-kubernetes/) May 1

Zero-downtime Deployment in Kubernetes with Jenkins (https://kubernetes.io/blog/2018/04/30/zero-downtime-deployment-kubernetes-jenkins/) Apr 30

Kubernetes Community - Top of the Open Source Charts in 2017 (https://kubernetes.io/blog/2018/04/25/open-source-charts-2017/) Apr 25

Kubernetes Application Survey 2018 Results (https://kubernetes.io/blog/2018/04/24/kubernetes-application-survey-results-2018/) Apr 24

Local Persistent Volumes for Kubernetes Goes Beta (https://kubernetes.io/blog/2018/04/13/local-persistent-volumes-beta/) Apr 13

Migrating the Kubernetes Blog (https://kubernetes.io/blog/2018/04/11/migrating-the-kubernetes-blog/) Apr 11

Container Storage Interface (CSI) for Kubernetes Goes Beta (https://kubernetes.io/blog/2018/04/10/container-storage-interface-beta/) Apr 10

Fixing the Subpath Volume Vulnerability in Kubernetes (https://kubernetes.io/blog/2018/04/04/fixing-subpath-volume-vulnerability/) Apr 4

(/)
Kubernetes 1.10: Stabilizing Storage, Security, and Networking (https://kubernetes.io/blog/2018/03/26/kubernetes-1.10-stabilizing-storage-security-networking/) Mar 26

Principles of Container-based Application Design (https://kubernetes.io/blog/2018/03/principles-of-container-app-design/) Mar 15

Expanding User Support with Office Hours (https://kubernetes.io/blog/2018/03/expanding-user-support-with-office-hours/) Mar 14

How to Integrate RollingUpdate Strategy for TPR in Kubernetes (https://kubernetes.io/blog/2018/03/how-to-integrate-rollingupdate-strategy/) Mar 13

Apache Spark 2.3 with Native Kubernetes Support (https://kubernetes.io/blog/2018/03/apache-spark-23-with-native-kubernetes/) Mar 6

Kubernetes: First Beta Version of Kubernetes 1.10 is Here (https://kubernetes.io/blog/2018/03/first-beta-version-of-kubernetes-1-10/) Mar 2

Reporting Errors from Control Plane to Applications Using Kubernetes Events (https://kubernetes.io/blog/2018/01/reporting-errors-using-kubernetes-events/) Jan 25

Core Workloads API GA (https://kubernetes.io/blog/2018/01/core-workloads-api-ga/) Jan 15

Introducing client-go version 6 (https://kubernetes.io/blog/2018/01/introducing-client-go-version-6/) Jan 12

Extensible Admission is Beta (https://kubernetes.io/blog/2018/01/extensible-admission-is-beta/) Jan 11

Introducing Container Storage Interface (CSI) Alpha for Kubernetes (https://kubernetes.io/blog/2018/01/introducing-container-storage-interface/) Jan 10

Kubernetes v1.9 releases beta support for Windows Server Containers (https://kubernetes.io/blog/2018/01/kubernetes-v19-beta-windows-support/) Jan 9

Five Days of Kubernetes 1.9 (https://kubernetes.io/blog/2018/01/five-days-of-kubernetes-19/) Jan 8

2017

2016

2015

Home (/docs/home/)

Blog (/blog/)

Partners (/partners/)

(/)

Community (/community/)

Case Studies (/case-studies/)

(https://twitter.com/kubernetesio) (https://github.com/kubernetes/kubernetes) (http://slack.k8s.io/)

(https://calendar.google.com/calendar/embed?

(http://stackoverflow.com/questions/tagged/kubernetes) (https://discuss.kubernetes.io) group.calendar.goog

Contribute (https://git.k8s.io/community/contributors/guide)