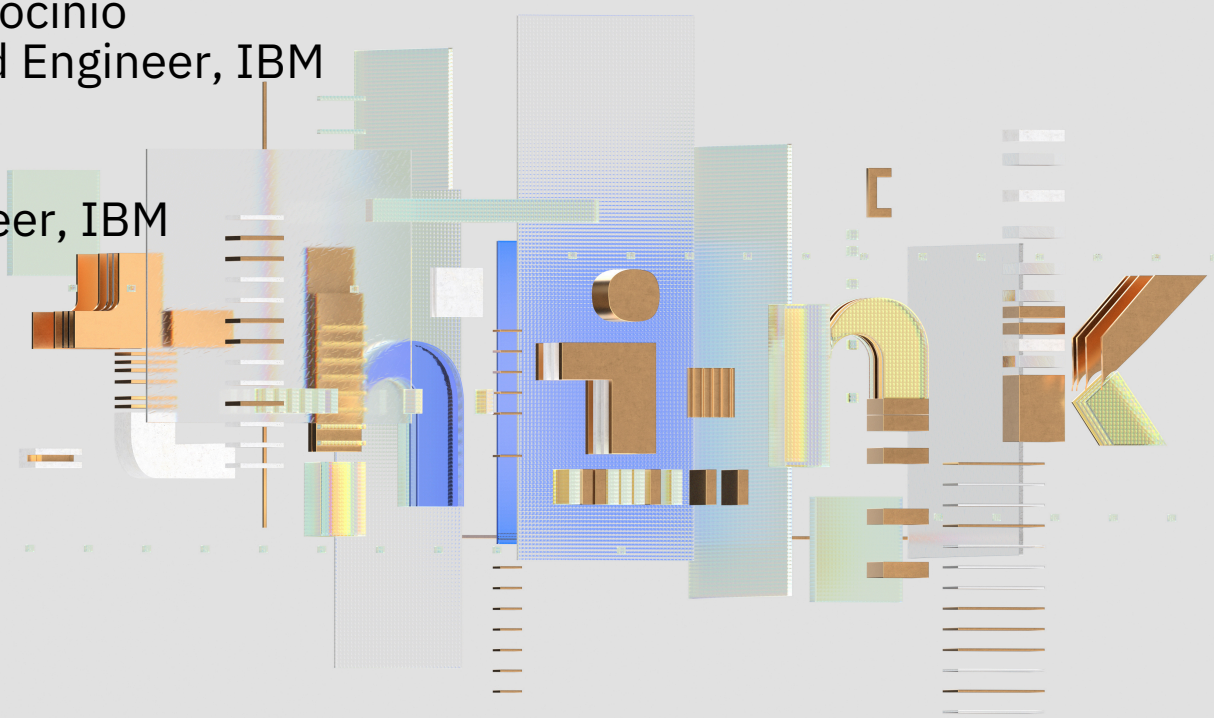
Kubernetes Basics with IBM Cloud Private

—

Eduardo Patrocinio
Distinguished Engineer, IBM

Sahdev Zala
Senior Engineer, IBM



think 2019

Contents

- Why Container Orchestration
- Kubernetes
  - Architecture
  - Resource Model
  - Pod
  - Deployment
  - Service
  - Application Deployment Workflow
  - Community Overview
- Helm
- Lab Overview

think

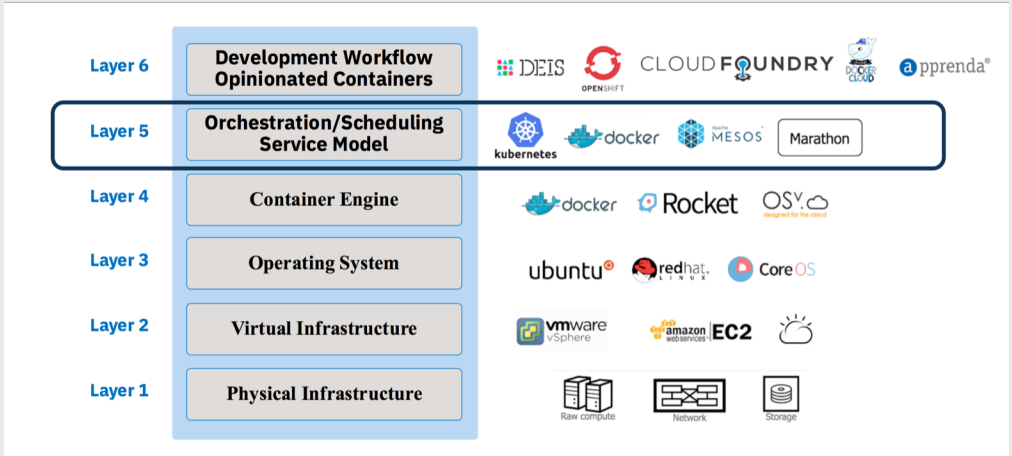
Containers Orchestration

- Automated deployment, scaling, and management of containerized applications
- Few to thousands of containers
- Kubernetes has emerged as the platform for containers orchestration
- IBM Cloud
  - IBM Cloud Private (ICP)
  - IBM Cloud Kubernetes Service (IKS)
  - Multi Cloud Manager
    - Single dashboard that lets enterprise oversee multiple Kubernetes clusters wherever they are—public cloud or private

Picture courtesy:
https://containerjournal.com/2017/05/25/forrester-rise-containers-forcing-automation/

What is Kubernetes?

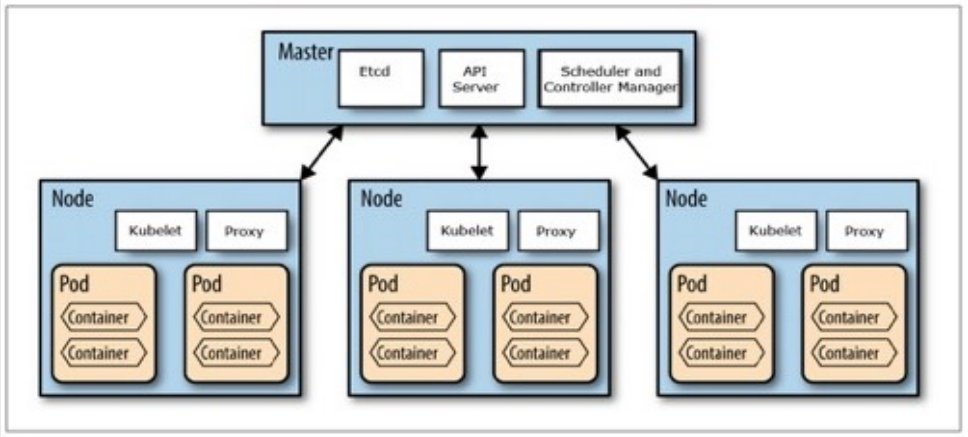
- Enterprise Level Container Orchestration
  - Provision, manage, scale applications (containers) across a cluster
  - Manage infrastructure resources needed by applications
    - Volumes, Networks, Secrets and many more
- Declarative model
  - User provide the "desired state" and Kubernetes will make it happen
- What's in a name?
  - Kubernetes (K8s/Kube): "Helmsman" in ancient Greek



- ✓ High Availability
  - ➢ Multiple replicas of app
- ✓ Portability
  - ➢ Runs across platforms
- ✓ Scalability
  - ➢ Policy (CPU/memory utilization) based scaling
- ✓ Security
  - ➢ TLS enabled endpoints, Secrets

Kubernetes Architecture

- Simple client-server model
- User request a specific state using client like *kubectl* or REST APIs
    - e.g. Deploy some app
- User request is watched and delegated to Controllers to take action
- Controllers take action to achieve the desired state



Kubernetes Resource Model

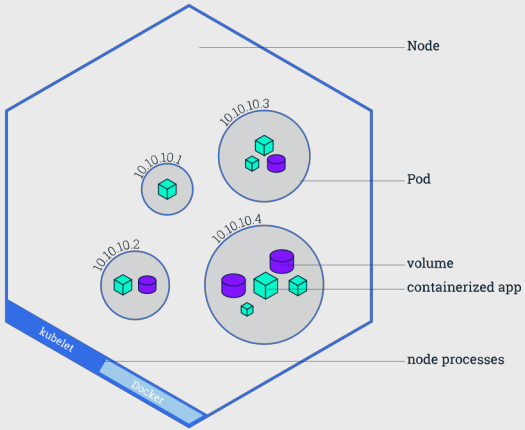
- Config Maps
- Daemon Sets
- **Deployments**
- Events
- Endpoints
- Ingress
- Jobs
- Nodes
- **Namespaces**
- **Pods**
- Persistent Volumes
- **Replica Sets**
- Secrets
- Service Accounts
- **Services**
- Stateful Sets, and more...

- Kubernetes does not have the concept of an application model, rather it defines a resource for every purpose
- User application is defined in Kubernetes by collection of these resources
- Each resource is monitored and processed by a controller

Pod

- Group of one or more containers, with shared storage/network, and a specification for how to run the containers
  - Smallest deployable unit in K8s
  - Not a durable entity, can die any time
  - User should not create Pod directly but with using resource like Deployment

- Pod always runs on a Node
  - A node is a worker machine in Kubernetes and may be a VM or physical machine

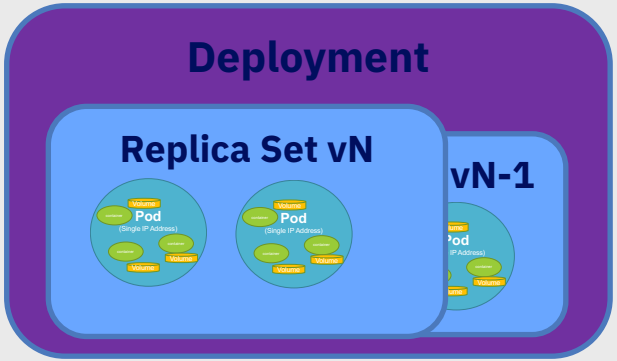



```
apiVersion: v1
kind: Pod
metadata:
  name: guestbook-pod
  labels:
    app: guestbook
spec:
 containers:
   - name: guestbook
     image: ibmcom/guestbook:v1
     ports:
       - containerPort: 3000
```

- Labels are key-value pairs that are attached to each object in Kubernetes
  - *kubectl get pods -l app=guestbook*

Deployment

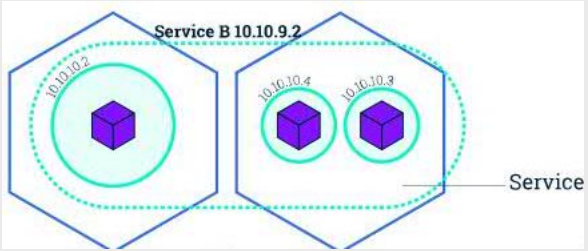
- A Deployment is responsible for creating and updating instances of your application
  - Manages Replica Sets
  - Manages Pods
- Deployment Controller continuously monitors those instances
  - Will create or delete Pods as needed to meet the replica count
- Deployments are also used to manage safely rolling out changes to your running Pods




```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: guestbook-deployment
  labels:
    app: guestbook
spec:
  replicas: 2
  selector:
    matchLabels:
      app: guestbook
  template:
    metadata:
      labels:
        app: guestbook
        version: v1
    spec:
      containers:
      - name: guestbook
        image: ibmcom/guestbook:v1
        ports:
        - containerPort: 3000
```

Service

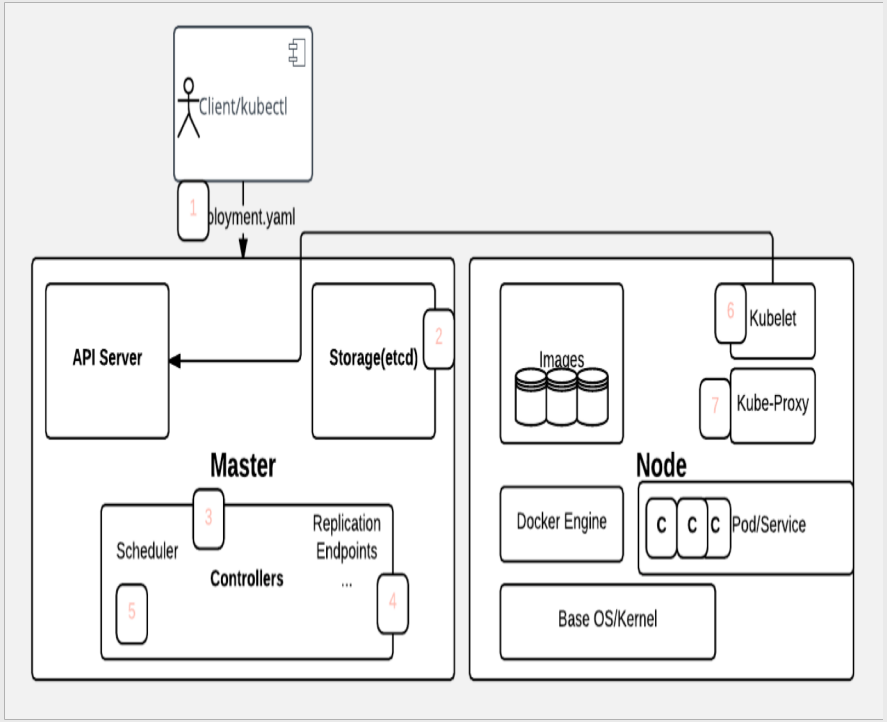
- Pod has unique IP but it’s not exposed outside the cluster, so how to allow your applications running in a Kubernetes cluster to receive traffic?
- A Service is an abstraction which defines a logical set of Pods running in your cluster, that all provide the same functionality
- Service is assigned with a unique IP address, which is tied to the lifespan of the Service
- Expose a Service onto an external IP address using NodePort, Load Balancer, Ingress Controller or Istio.




```
apiVersion: v1
kind: Service
metadata:
  name: guestbook
  labels:
    app: guestbook
spec:
  ports:
  - port: 3000
    targetPort: http-server
  selector:
    app: guestbook
  type: NodePort
```

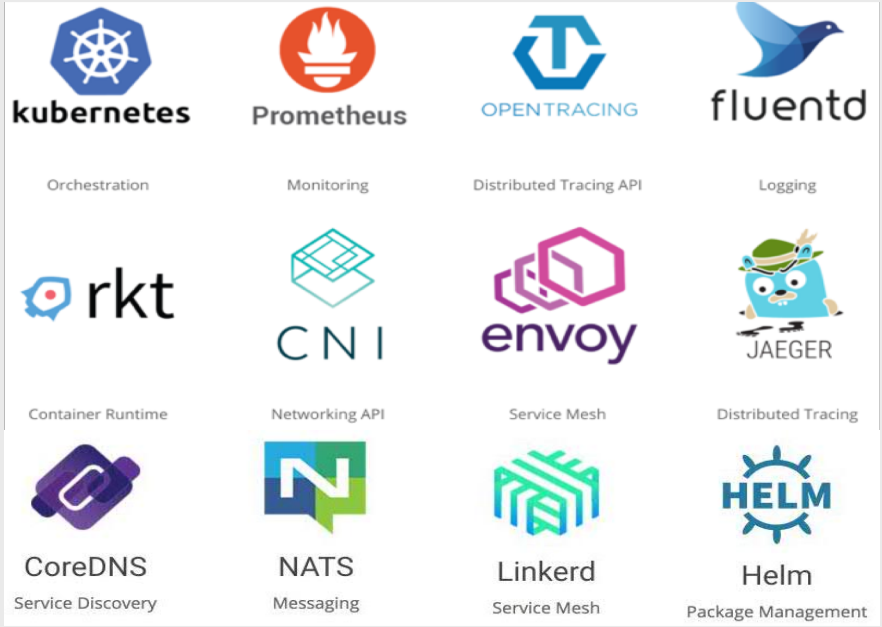
Kubernetes Application Deployment Workflow

- **Kubectl** - user deploys a new application via Kubectl. It sends the request to the API Server.
- **API server** - receives the request and stores it in the data store (etcd). User request is watched and delegated to Controllers to take action.
- **Controllers** - detects the new request and match the desired state.
  - e.g. Scheduler assigns new pod to a Node
- **Kubelet** - takes a set of PodSpecs and
- ensures that the containers described in those PodSpecs are running and healthy.
- **Kubeproxy** - manages network traffic for the pods – including service discovery and load-balancing.



Kubernetes Community Overview

- Kubernetes is a CNCF project
- What is CNCF?
  - **C**loud **N**ative **C**omputing **F**oundation
- Various Repositories
  - main - github.com/kubernetes/Kubernetes
  - doc - github.com/kubernetes/website
  - community - github.com/kubernetes/community
- Issues https://github.com/kubernetes/kubernetes/issues
- Pull Requests https://github.com/kubernetes/kubernetes/pulls



Where is the Kubernetes Community?

- Special Interest Groups (SIGs)
  - Domain specific group
    - Examples: ContribEx SIG, Networking SIG, Testing SIG, Doc SIG, **SIG IBM Cloud** etc.
    - https://github.com/kubernetes/community/blob/master/sig-list.md
  - Meet every week or bi-weekly
    - Typically zoom video call
    - Similar to OpenStack IRC project meetings
- Slack channels
  - https://kubernetes.slack.com
  - Several channels for users, developers, SIGs
- Weekly Community Meeting
- Conferences (CloudNativeCon + KubeCon) / Meetups
- Mailing list / google groups:
  - Devs: kubernetes-dev@googlegroups.com
  - Users: kubernetes-users@googlegroups.com

Helm

- How to deploy your app without managing the individual manifest files/resources?
- How to update it without having to update manifest files and redeploy?
- How to share deployment artifacts with others?
- Helm can answer these questions
  - Helm is a package manager that streamlines installation and management of Kubernetes applications.
  - Think of it like apt/yum/homebrew for Kubernetes.
  - It is a CNCF project and its development is backed by a strong open community.

Create and Manage Kubernetes Resources with *kubectl*

Inline

- *kubectl run guestbook --image=ibmcom/guestbook:v1*
- kubectl expose deployment guestbook --type="NodePort" --port=3000

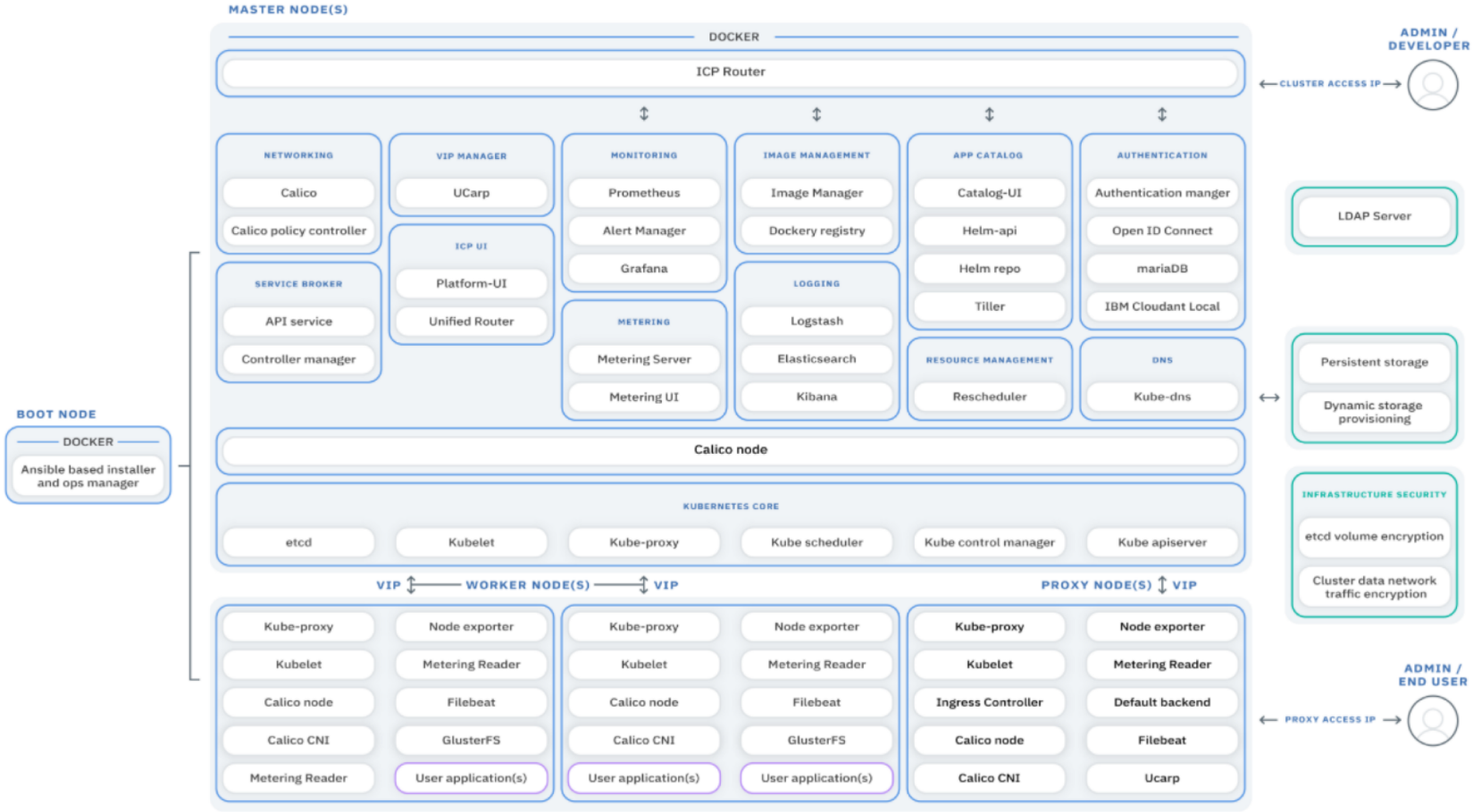
➢ Simple and quick but doesn’t give much control
➢ OK for playing but not for production
  - Commands do not provide a source of records except for what is live.
  - Commands do not provide a template for creating new objects
  - etc.

Using YAML manifest

- *kubectl create -f guestbook-deployment.yaml*
- *kubectl create -f guestbook-service.yaml*

➢ Requires basic understanding of the object schema and the additional step of writing a YAML file but,
  - Allows you to have more fine-grained control over all of resources being created within the Kubernetes cluster
  - Reusable and easy to share
➢ **This is what we will use for labs** except Lab1 where we will create namespace to show inline command usage.

IBM Cloud Private (ICP)



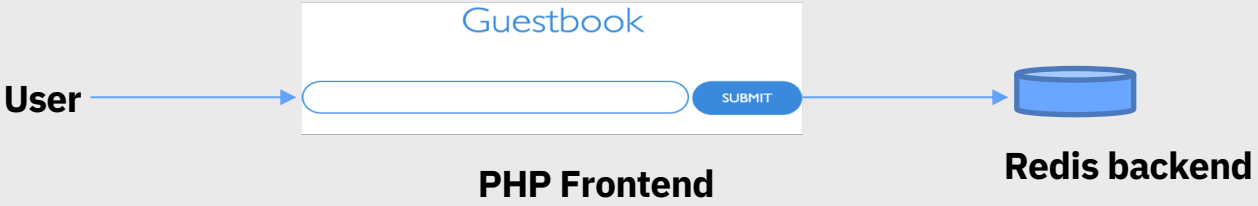
Lab Structure

- Namespace and Context
- Create Deployment
- Scale the Deployment
- Create Service
- Expose Service externally
- Run Troubleshooting Commands
- Deploy App using Helm
- Create Environmental Variable to use in the container
- Create ConfigMap
- Create Secret

Follow the instructions provided here:
http://ibm.biz/icpkube

We will use guestbook app

- Sample multi-tier web application which stores guest entries
- Make sure you look into the lab files and understand what's going on



Lab

http://ibm.biz/icpkube

Notices and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

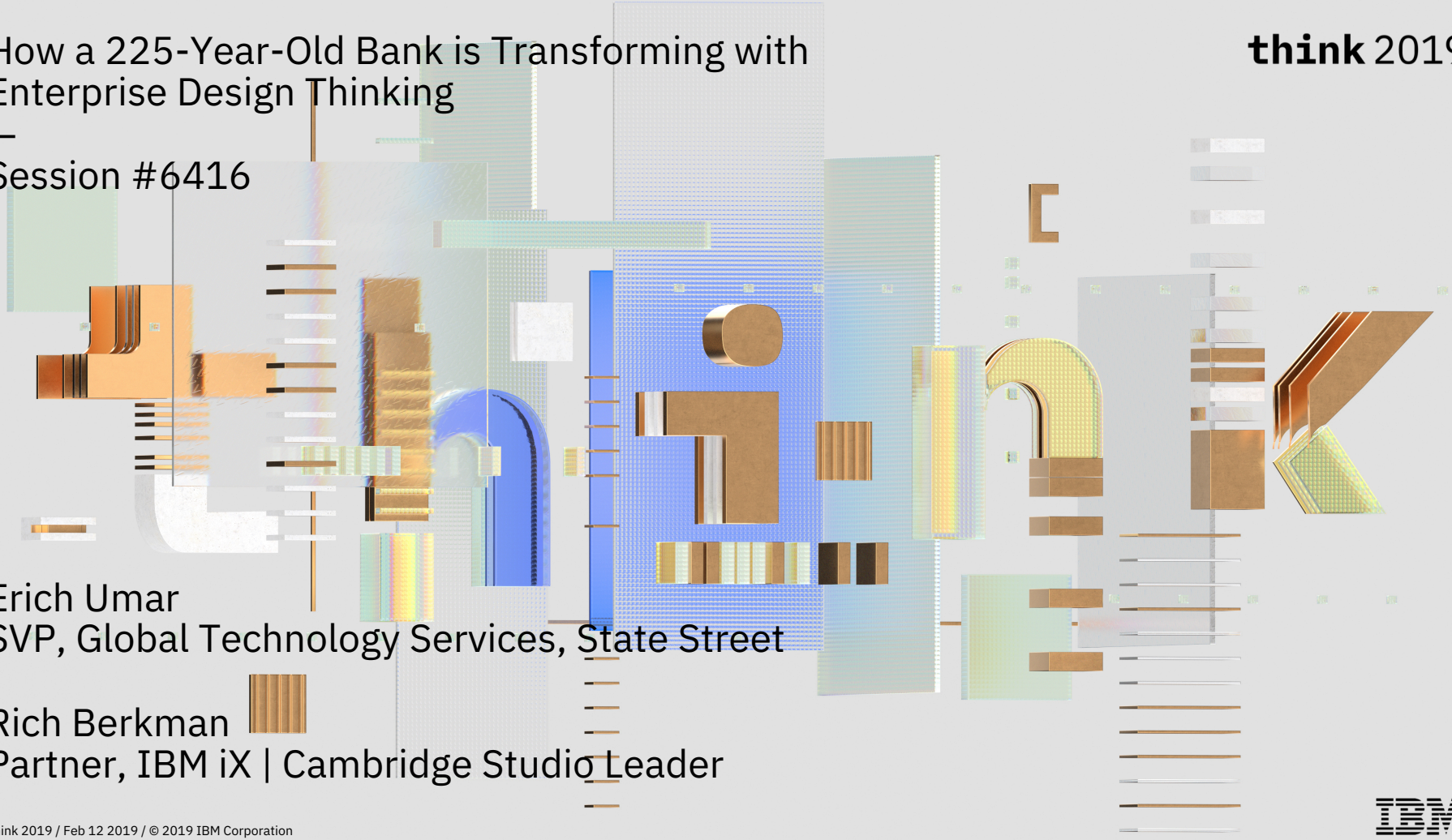
IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: .

How a 225-Year-Old Bank is Transforming with Enterprise Design Thinking

—

Session #6416

think 2019



Erich Umar
SVP, Global Technology Services, State Street

Rich Berkman
Partner, IBM iX | Cambridge Studio Leader

IBM

Thank you

Eduardo Patrocinio
Distinguished Engineer, IBM

Sahdev Zala
Senior Engineer, IBM

—
eduardop@us.ibm.com
spzala@us.ibm.com

Back Up