



Running game servers on Kubernetes

Mo Firouz

About me

- My name is Mo(hammad) Firouz!
- Originally Iranian - can speak Farsi with a funny accent.
- Software architect
- Co-founder of Heroic Labs

Heroic Labs

- Nakama, a realtime and social server for games.
- Completely open-source (and free!):
<https://github.com/heroiclabs/nakama>
- Realtime, server-authoritative multiplayer
- Flexible matchmaking system
- Extendable using Lua scripting
- Friends, Clans, Chat + more



Nakama



Things to discuss

1. Intro to Kubernetes
2. System overview
3. Two examples of game servers
 - Nakama server
 - Unity Headless servers

Intro to Kubernetes

*System for automating deployment, scaling, and management of **containerized** applications*

- Hardware provisioning - replaces Puppet, Chef and Ansible.
- Supervisor-like responsibilities - healthcheck and uptime
- Log aggregation + metrics + scheduled jobs + (much) more



Intro to Kubernetes

Some rules to follow:

1. Log to stdout and stderr
2. Return non-zero for abnormal exits
3. Add healthcheck endpoint - ideally to /
4. Add readiness endpoint - ideally to same as healthcheck
5. Use cmd args and env variables, not config files

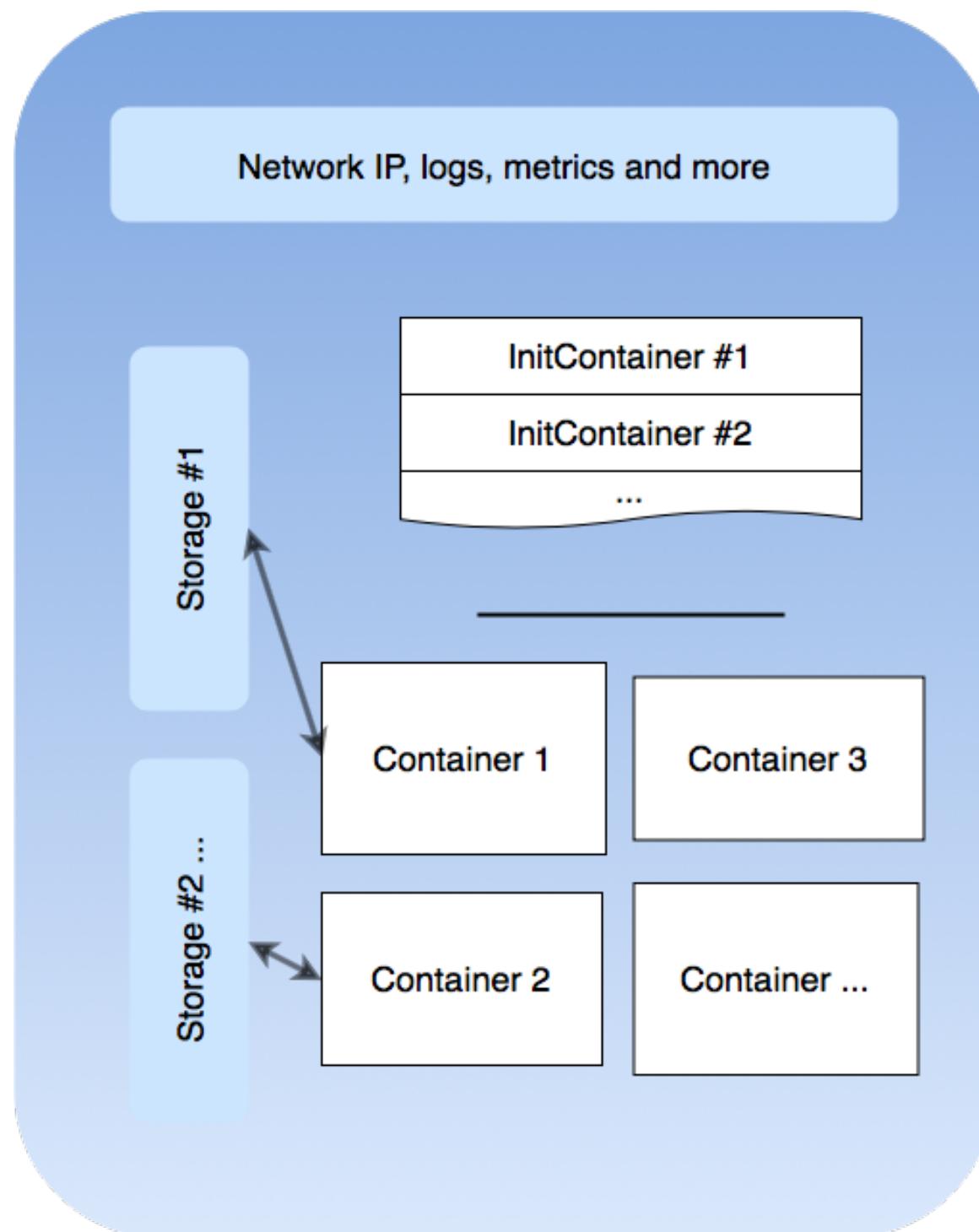
Intro to Kubernetes

Kubernetes components

1. Pod
2. Controllers
3. Service + Ingress
4. Nodes
5. Autoscaling

Pod

- Wrap one or more containers
- Pods are ephemeral
 - can be dynamically created / deleted
- Unique network cluster-IP (but not permanent)
- Storage resource(s)





Pod

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nakama
spec:
  volumes:
  - name: nakama-volume
    emptyDir:
  containers:
  - name: nakama
    image: heroiclabs/nakama
    command: "/bin/sh exec nakama"
    ports:
    - containerPort: 7350
      name: api
  volumeMounts:
  - mountPath: /nakama-data
    name: nakama-volume
```

Service

- A way to access pods
- Services have DNS names
 - controller.namespace.service.cluster.local
 - controller.namespace
- Expose applications internally and externally

Service

- ClusterIP: cluster-internal IP
- NodePort: Expose port (mapping) on the physical node
- LoadBalancer: Cloud provider's load balancer
- ExternalName: CNAME without any proxying



Service

```
apiVersion: v1
kind: Service
metadata:
  name: nakamaservice
  labels:
    service: nakama
spec:
  selector:
    app: nakama
  type: NodePort
  ports:
  - port: 80
    targetPort: 7350
    name: api
```

Ingress

- Very similar to LoadBalancer service
- Only allowing HTTP(S) traffic into the cluster
- SSL Termination
- Usually configures Cloud Provider traffic

Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nakamaingress
  labels:
    ingress: nakama
  annotations:
    kubernetes.io/ingress.allow-http: "false"
spec:
  tls:
    - secretName: ingress-cert-secret
  backend:
    serviceName: nakama
    servicePort: 80
```

Controllers

- Replicating homogeneous set of pods
- Ensures healthy pods are running
- Deployment:
 - Manage infrastructure state from current to desired
 - Rolling update
 - Scale up/down pod replicas
 - History and rollback

Controller

- StatefulSet:
 - Very similar to Deployment
 - Sticky pod IDs - sequential gurantueed ordering
 - Disk and Network resources will stay the same across roll outs
- DaemonSets, (Cron)Jobs, and more...



Controller

```
apiVersion: apps/v1beta2
kind: StatefulSet
metadata:
  name: nakamastatefulset
spec:
  serviceName: "nakamaservice" # This must exist before Statefulset is created
  revisionHistoryLimit: 1
  replicas: 2
  podManagementPolicy: OrderedReady # or Parallel
  updateStrategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: nakama # this must matches below
  template:
    metadata:
      labels:
        app: nakama
# ...rest of pod template...
```

Nodes

- Assign special pods to special nodes

```
nodeSelector:  
  disktype: ssd
```

- Use affinity and anti-affinity to indicate soft/hard requirement for node attractiveness
- Use taint and tolerations to repel pods from nodes
- Use both to schedule pods only on nodes with special hardware
- Ensure to use resource requests and limits for pods to limit noisy neighbour problems.

Nodes

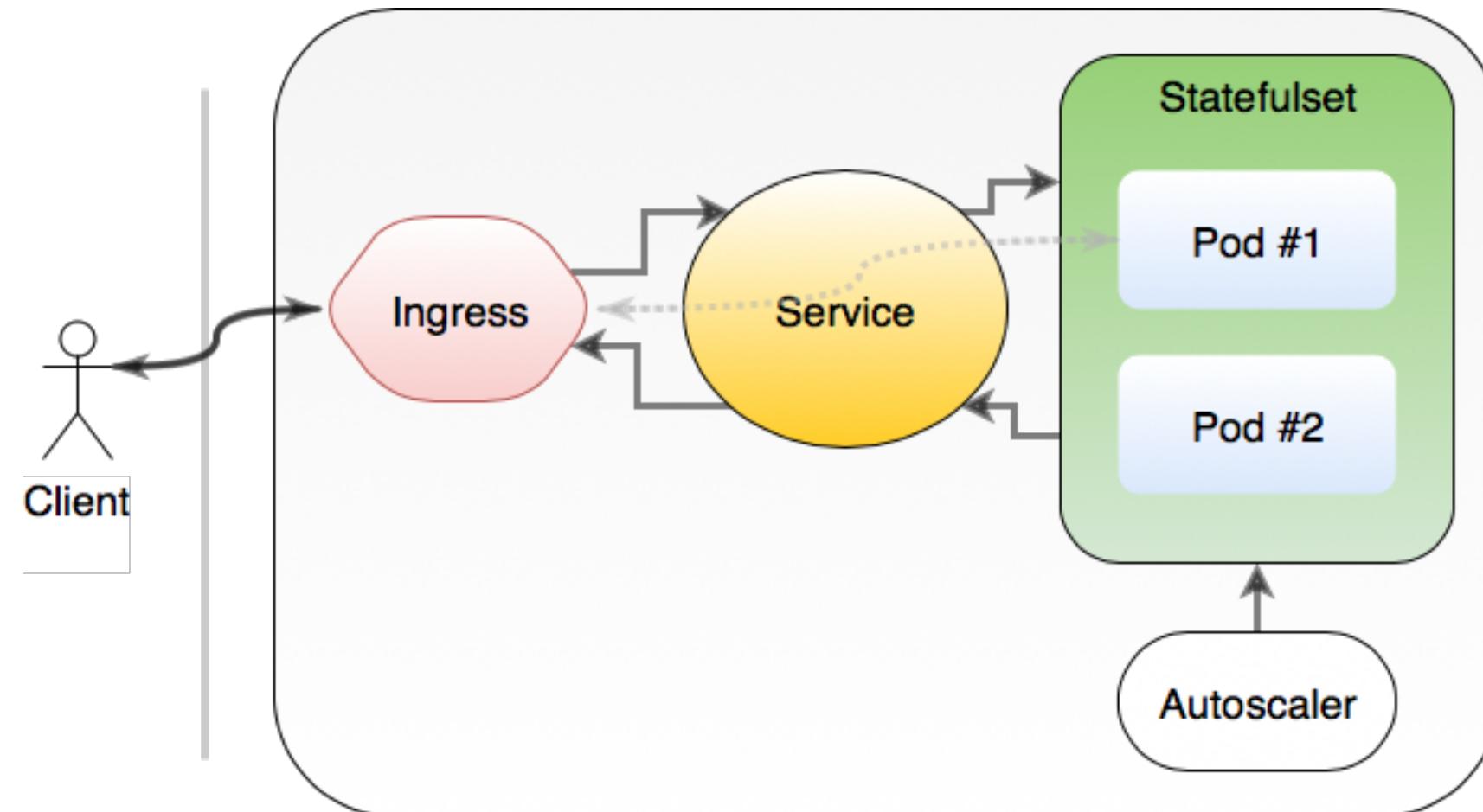
```
affinity:  
podAntiAffinity:  
  preferredDuringSchedulingIgnoredDuringExecution:  
    - weight: 100  
      podAffinityTerm:  
        topologyKey: kubernetes.io/hostname  
        labelSelector:  
          matchExpressions:  
            - key: app  
              operator: In  
              values:  
                - nakama
```



Autoscaling

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: nakamascaler
  labels:
    autoscaler: nakama
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: StatefulSet
    name: nakamastatefulset
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      targetAverageUtilization: 50
      name: cpu
```

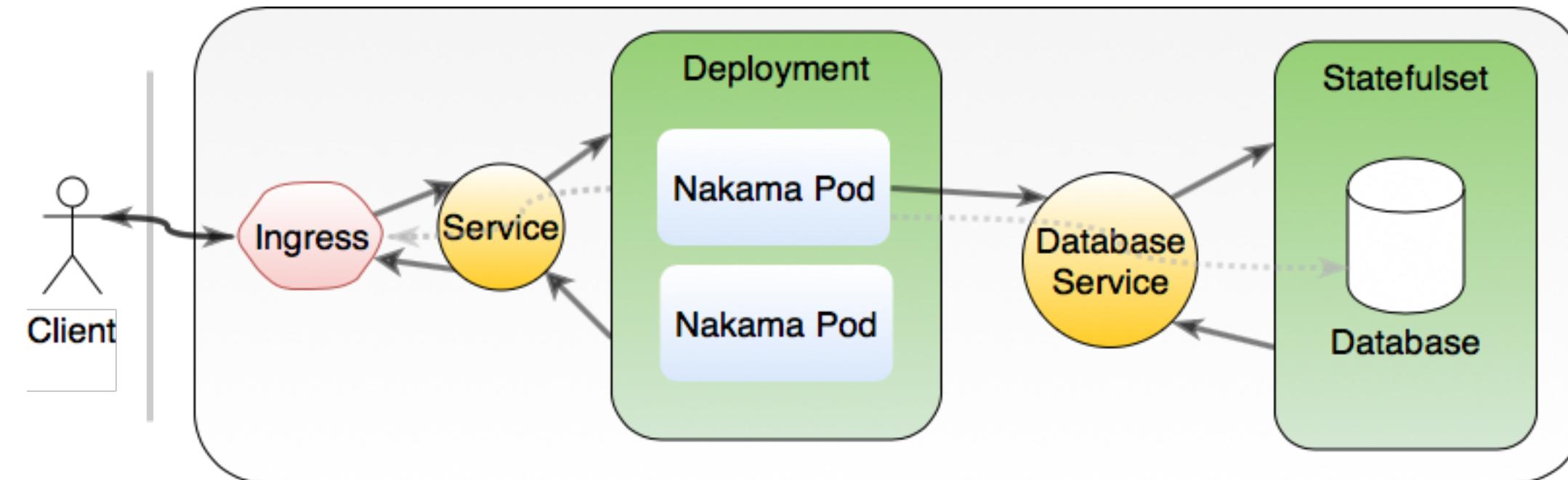
System overview



Nakama game server

- Realtime multiplayer
- Server-authoritative multiplayer
- Matchmaking
- Custom code runtime
- Presence system
- Friends, Groups/clans
- Realtime chat, notifications
- User accounts, authentication

Nakama game server



Nakama game server

Pods							
Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)	
 nakama-0	nakama-n1s1-7a45d00d-b0n3	Running	0	10 hours	 0.005	 64.531 Mi	
 cockroachdb-1	db-n1s2-a7f0d49af90p	Running	0	16 days	 0.019	 1.016 Gi	
 cockroachdb-0	db-n1s2-a7f0d49a-8cb2	Running	0	16 days	 0.016	 973.879 Mi	
 cockroachdb-2	db-n1s2-a7f0d49a-4ts8	Running	0	16 days	 0.017	 1,013.605 Mi	

Stateful Sets				
Name	Labels	Pods	Age	Images
 nakama	statefulset: nakama	1 / 1	a month	heroiclabs/nakama:2.0.1
 cockroachdb	statefulset: cockroachdb	3 / 3	2 months	cockroachdb/cockroach:v2.0.1

Nakama game server

Ingresses					
Name	Endpoints	Age			
nakama	35.186.228.0	2 months			

Services					
Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
<input checked="" type="checkbox"/> nakama	service: nakama	10.23.250.87	nakama.tgc:7350 TCP nakama.tgc:30570 TCP	-	a month
<input checked="" type="checkbox"/> cockroachdb	service: cockroachdb	None	cockroachdb.tgc:26257 T cockroachdb.tgc:0 TCP cockroachdb.tgc:8080 TC cockroachdb.tgc:0 TCP	-	2 months



Unity headless server

```
/Applications/Unity/Unity.app/Contents/MacOS/Unity -quit -batchmode -nographics \
LogFile .unity-build.log \
-projectPath $(pwd)/unity-fastpacedmultiplayer \
-executeMethod BuildTools.QuickBuildLinux
```

```
public static void QuickBuildLinux()
{
    BuildPlayerOptions opts = new BuildPlayerOptions
    {
        options = BuildOptions.Development & BuildOptions.EnableHeadlessMode,
        locationPathName = "/Users/mo/Desktop/server",
        target = BuildTarget.StandaloneLinux64
    };

    BuildPipeline.BuildPlayer(opts);
}
```



Unity headless server

```
metadata:  
  generateName: "unity-"  
labels:  
  app: unity-server  
spec:  
  hostNetwork: true  
  containers:  
    - image: mofirouz/unity-fastpacedmultiplayer:0.0.1  
      name: unity-server  
      imagePullPolicy: Always  
      command:  
        - "/bin/sh"  
        - "-ecx"  
        - >  
          exec ./server -logFile /dev/stdout  
  ports:  
    - containerPort: 7777  
      name: server-api
```



Feel free to send questions;

email: mo@heroiclabs.com

twitter: [@mofirouz](https://twitter.com/@mofirouz)

skype: mo.firouz

Talk available on GitHub:

<https://github.com/mofirouz/gameservers-kubernetes>



WWW.TEHRANGAMECON.COM