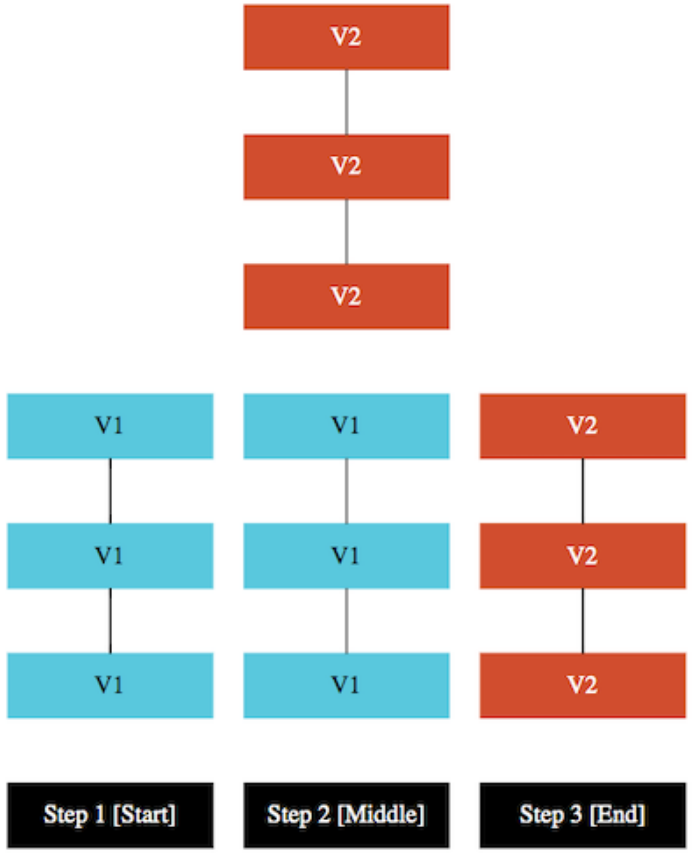
Release Management

Release Management

In28 Minutes

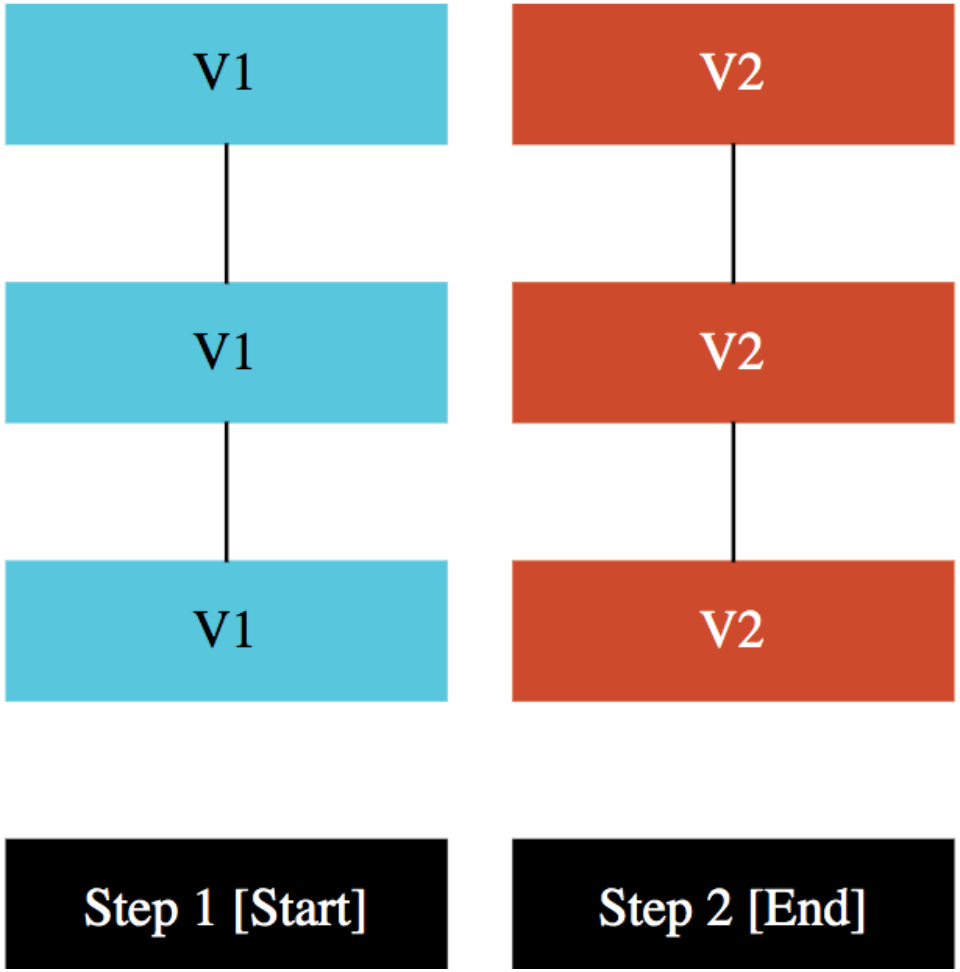
- **Goals**: vary from app to app
  - Zero Downtime
  - Only one version live at a time
  - Minimize Costs (and infrastructure needed)
  - Test with production traffic before going live
- **Best Practices**:
  - Small incremental changes
  - Automation (as much as possible)
  - Handling problems with new releases:
    - Analyze logs and metrics from Cloud Monitoring and Logging
    - Rollback to previous release and try replicating the problem in other environments



Deployment Approach : Recreate

In28 Minutes

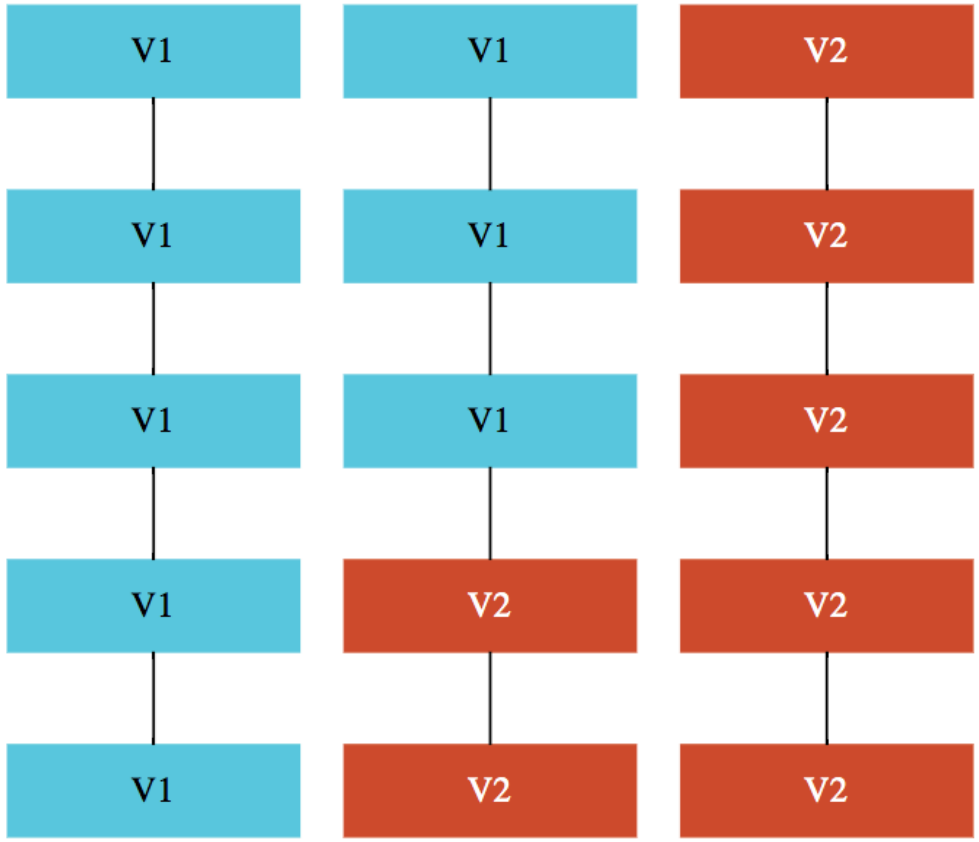
- **Approach:**
  - Terminate Version 1
  - Roll out Version 2
- **Characteristics:**
  - App down during the release
  - Rollback needs redeployment
    - AND more downtime
  - Cost effective and Fast
    - BUT disruptive
  - Avoid need for backward compatibility (data and applications)



Deployment Approach : Canary

In28 Minutes

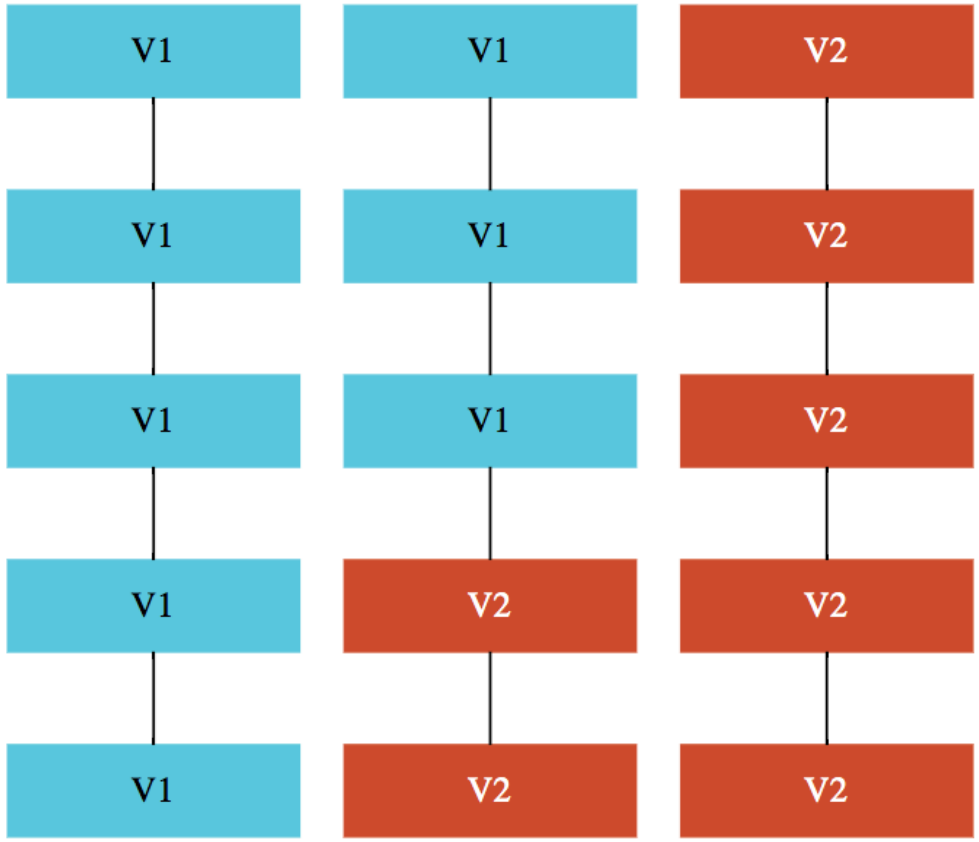
- **Approach:**
  - **Step 1:** V2 rolled out to a subset of instances
  - **Step 2:** On successful testing, V2 rolled out to all instances
    - OR V2 is rolled back in case of failure
- **Characteristics:**
  - Fast
  - Zero downtime
  - No extra infrastructure
  - Minimizes impact to users (in case of release failures)
  - Needs Backward compatibility (data and applications)



Testing Approach : A/B Testing

In28 Minutes

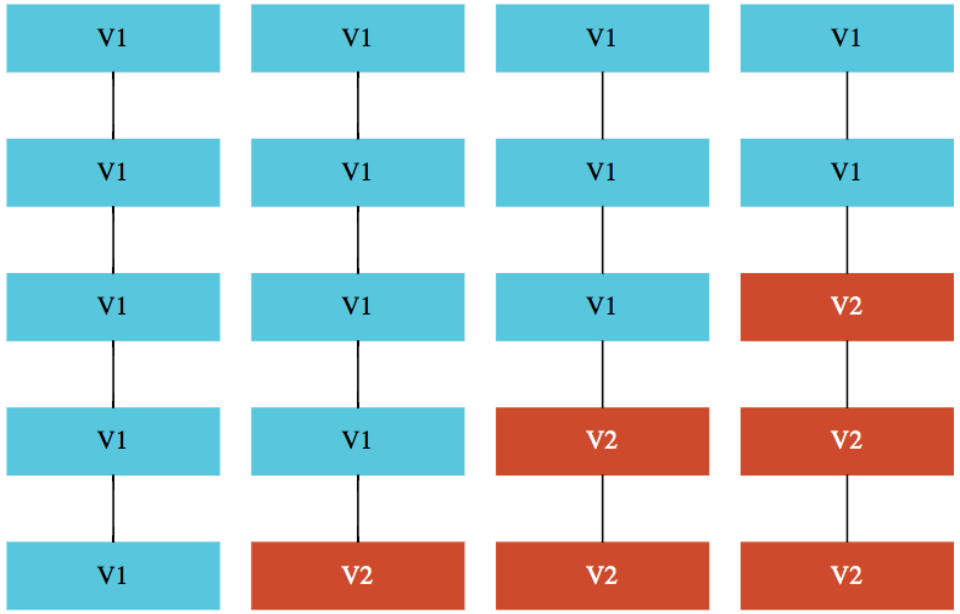
- **Use case**: You want to see if users like a feature!
- **Approach:**
  - **Step 1**: V2 (with new feature) rolled out to a subset of users
  - **Step 2**: On successful testing, V2 rolled out to all users
    - OR we go back to V1 in case users don't like the feature!
- **Characteristics:**
  - Gives the ability to test if your users like a feature



Deployment Approach : Rolling

In28 Minutes

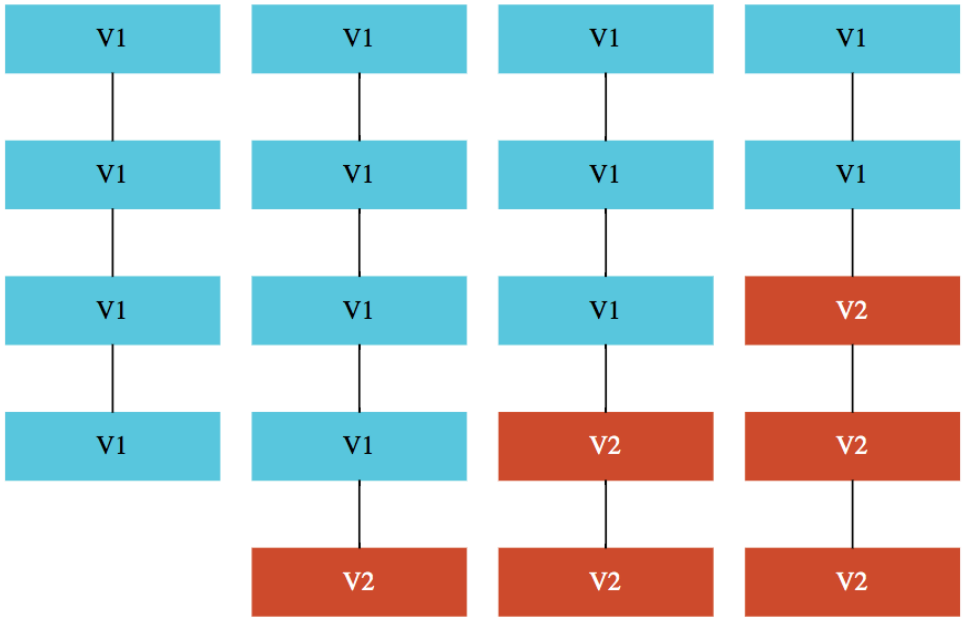
- **Approach:**
  - **Step 1**: V2 rolled out to a percentage of instances (Example window size: 5%)
  - **Step 2..N**: V2 gradually rolled out to rest of the instances (Example: 5% at a time)
- **Characteristics:**
  - Slow
  - Zero downtime
  - Needs automation and additional setup
  - No extra infrastructure
  - Minimizes impact to users (in case of release failures)
  - Needs Backward compatibility (data and applications)



Deployment Approach : Rolling with Additional Batch

In28 Minutes

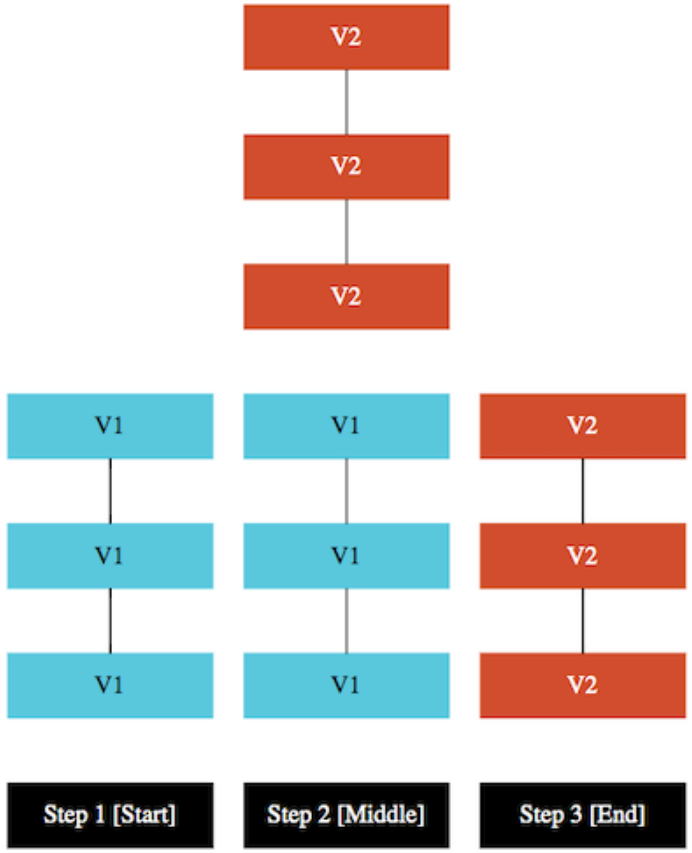
- **Approach:**
  - **Step 1**: Additional batch of new instances are created with V2 (Example: 5%)
  - **Step 2..N**: V2 gradually rolled out to the instances batch by batch (Example: 5% at a time)
- **Characteristics:**
  - Same as Rolling Deployment except for:
    - Needs Little bit of extra infrastructure
    - ZERO reduction in number of instances handling user requests



Deployment Approach : Blue Green

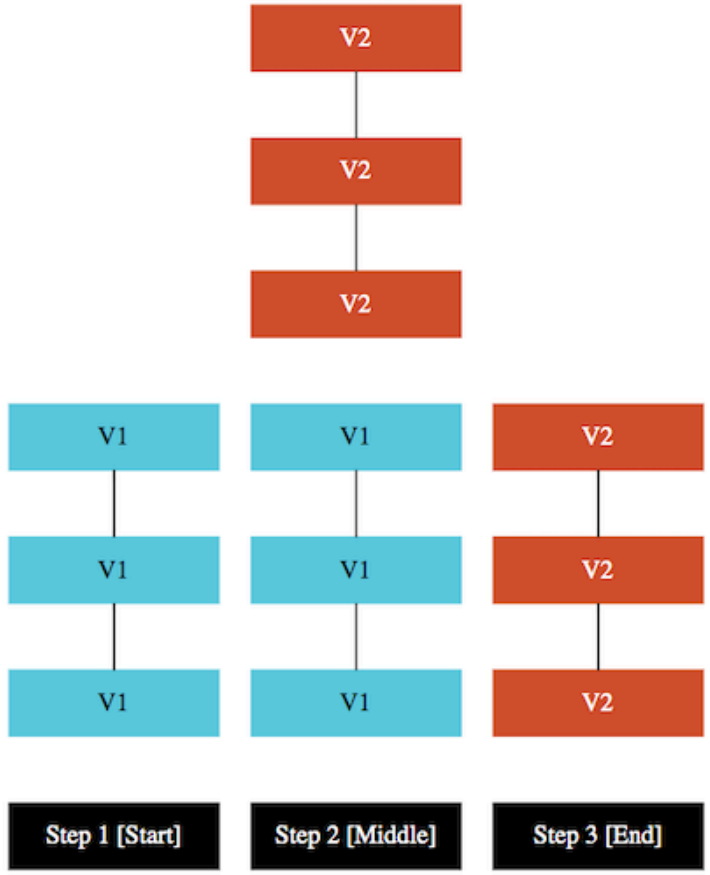
In28 Minutes

- **Approach:**
  - **Step 1:** V1 is Live
  - **Step 2:** Create (or replicate) a parallel environment with V2
  - **Step 3:** Switch all traffic from V1 to V2 (and remove V1 Environment)
- **Characteristics:**
  - Instant
  - Zero Downtime
  - Easy Rollback
  - Needs additional infra (during the release)
  - ZERO reduction in available capacity
  - Needs Backward compatibility (data and apps)



Testing Approach : Shadow

- **Approach:**
  - **Step 1**: V1 is Live
  - **Step 2**: Create (or replicate) a parallel environment with V2
    - Mirror traffic to V1 and V2
  - **Step 3**: Switch all traffic from V1 to V2 (and remove V1 Environment)
- **Characteristics:**
  - Zero production impact: Test V2 with real production traffic before releasing
    - You can also capture and replay live production traffic
  - Complicated : You don't want double payments (might need stubbing)
  - Needs a lot of additional infrastructure



Deployment Approaches - Managed instance group (MIG)

In 28 Minutes

Option	Details
Rolling Release	*gcloud compute instance-groups managed* ***rolling-action start-update*** *my-mig --version=template=v2-template* --max-surge=5 or 10% (Max instances updated at a time) --max-unavailable=5 or 10% (Max instances that can be down during update)
Canary Release	gcloud compute instance-groups managed **rolling-action start-update** my-mig --version=template=v1-template --canary-version=template=v2-template,target-size=10%
Blue Green Deployment	Create a new MIG and make manual adjustments to Load Balancer backends as needed

App Engine - Releasing New Versions

In28 Minutes

Option	Details
Deploy & shift all traffic at once	gcloud app deploy
Deploy v2 without shifting traffic	--no-promote
Shift traffic to V2 all at once	gcloud app services set-traffic s1 --splits V2=1
Gradual Migration	Add `--migrate` option to previous command
A/B testing	gcloud app services set-traffic s1 --splits=v2=.5,v1=.5

GKE - Releasing New Versions

In28 Minutes

Option	Details
Recreate	Set `strategy > type` on Deployment to `Recreate` Use 'kubectl set image deployment' OR Update deployment YAML to perform deployment
Rolling Update	Set `strategy > type` on Deployment to `RollingUpdate` Use 'kubectl set image deployment' or Update deployment YAML to perform deployment Configure `maxSurge` and `maxUnavailable` to control deployment
Blue Green Deployment	Create New Deployment. Control traffic using Ingress (or Service)
Canary Deployment	Needs Service Mesh like Istio