Meetup K8s Dont's

- De kantoor sessie

Brief introduction

- Marco Verleun
  - Older than the internet
  - Even older than Unix… ;-)
- Engineer at SUE
- Currently working as DevOps engineer for a government agency:
  - Gitlab
  - Kubernetes clusters
  - Many legacy applications

Topics

- Cluster sizing
- Resources, requests and limits
- Swap...
- Pressure
- Probes

Prepare

- Zip download: https://github.com/mverleun/Meetup-Kubernetes-Donts/archive/master.zip
- git clone https://github.com/mverleun/Meetup-Kubernetes-Donts.git

Cluster sizing

- Which is better?:

  A. More nodes with fewer resources each?

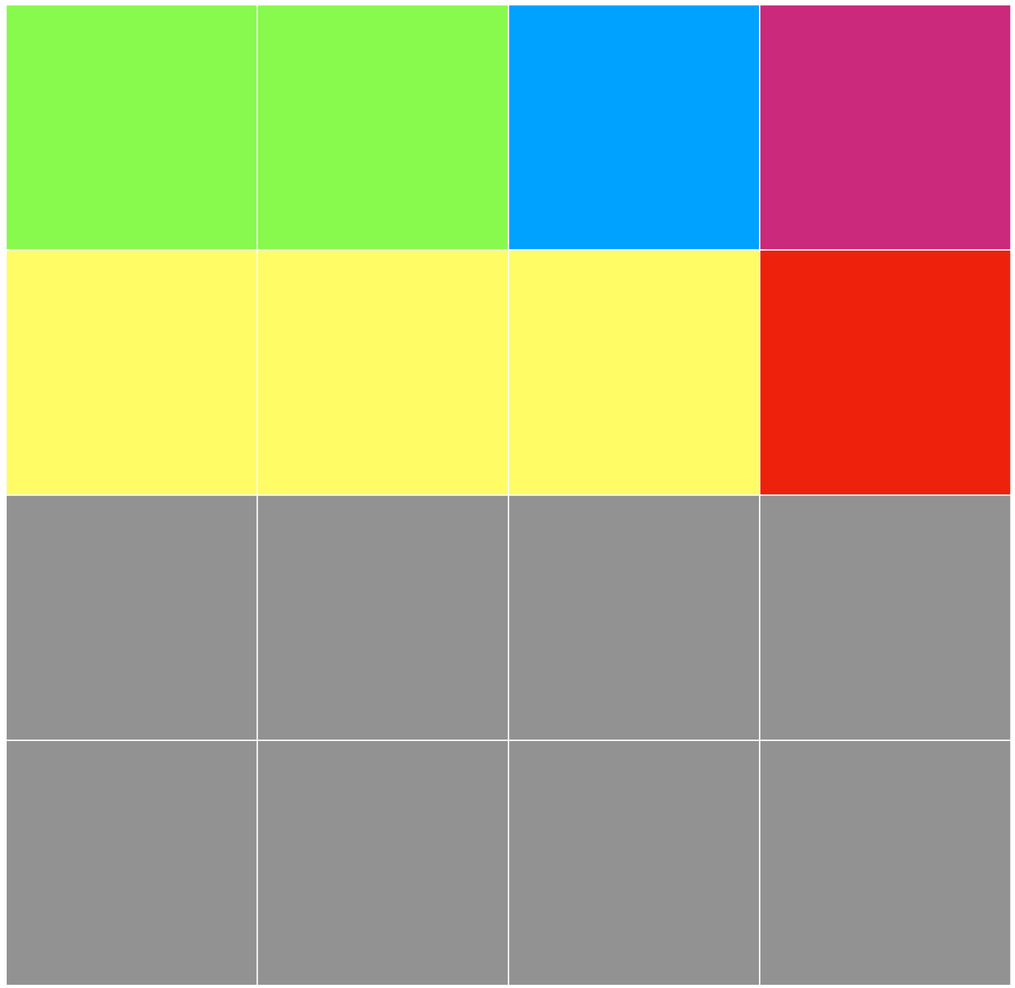
  B. Fewer nodes with more resources each?

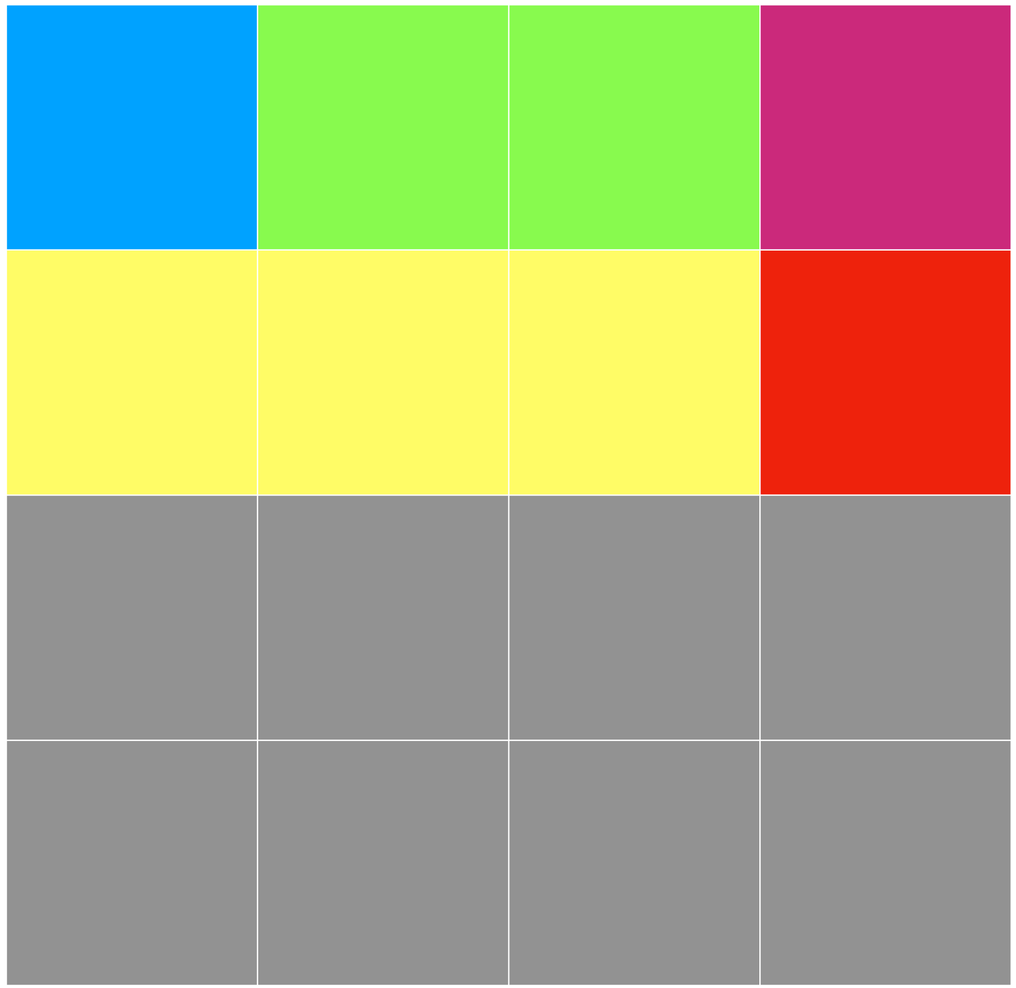
  C. Both are equal?

Kubertetris

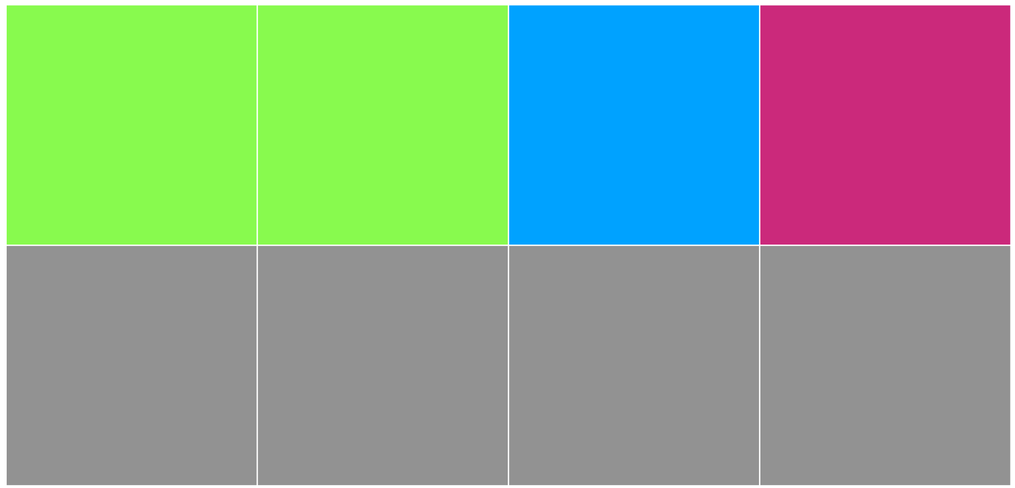
- Let’s play Kubertetris and see…
- Just like Tetris we have to make sure that we can deploy somewhere on the cluster.
- We have two clusters. Both have an equal amount of resources but the number of nodes differ.
- We’ll try to deploy the same applications in the same way.

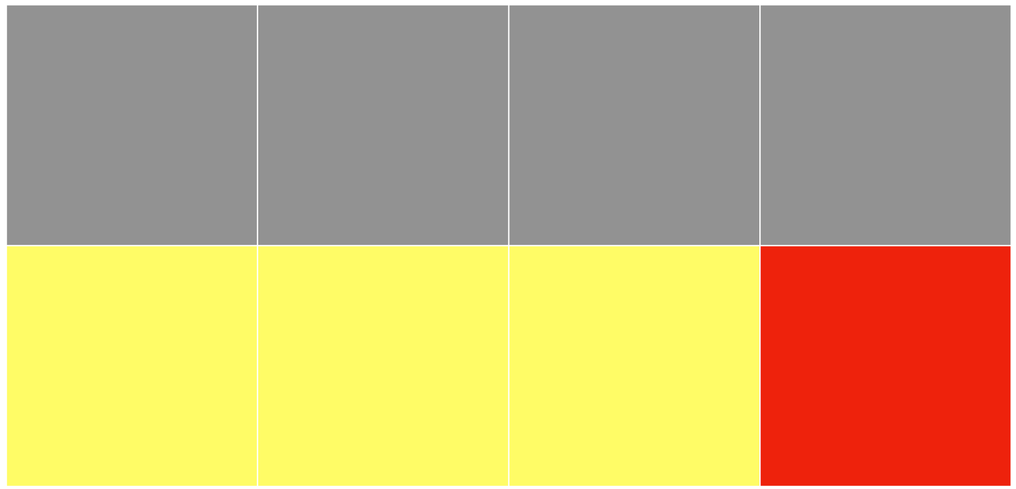
Two nodes, preloaded

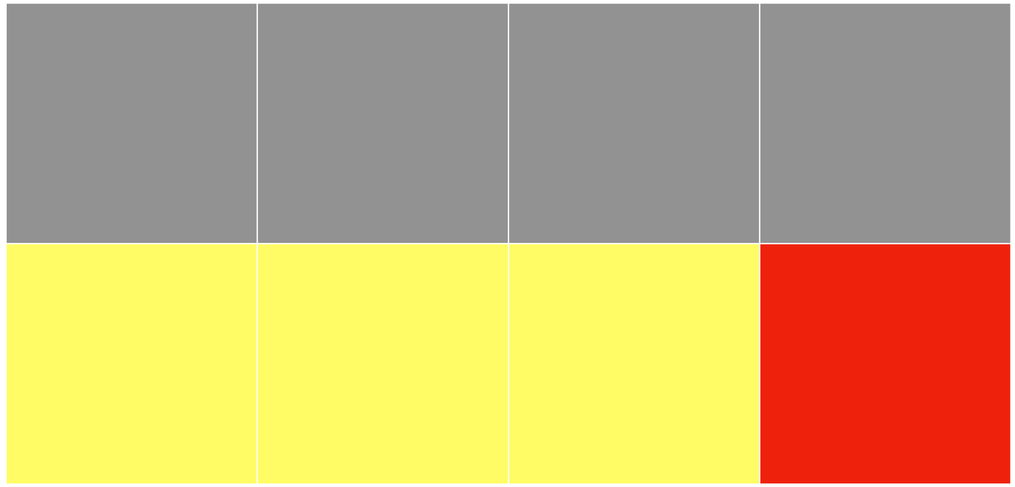


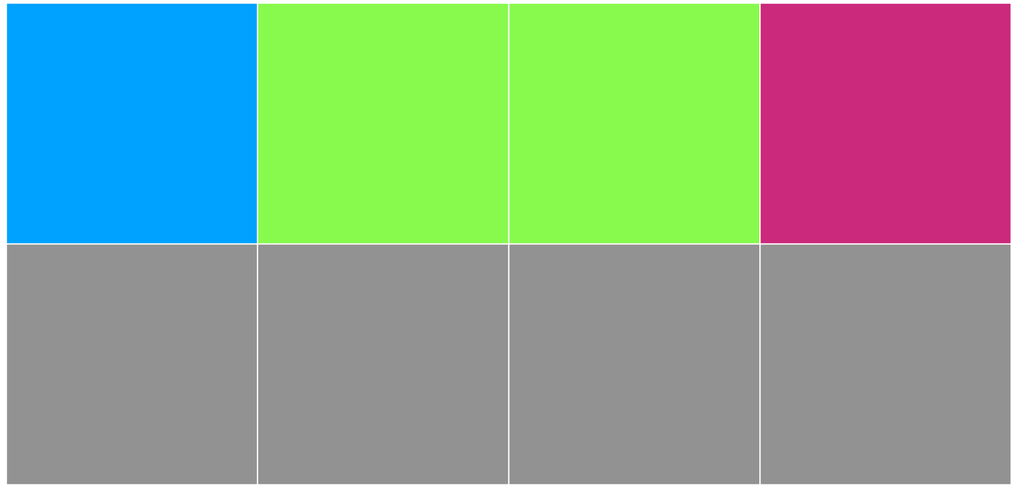


Four nodes, preloaded





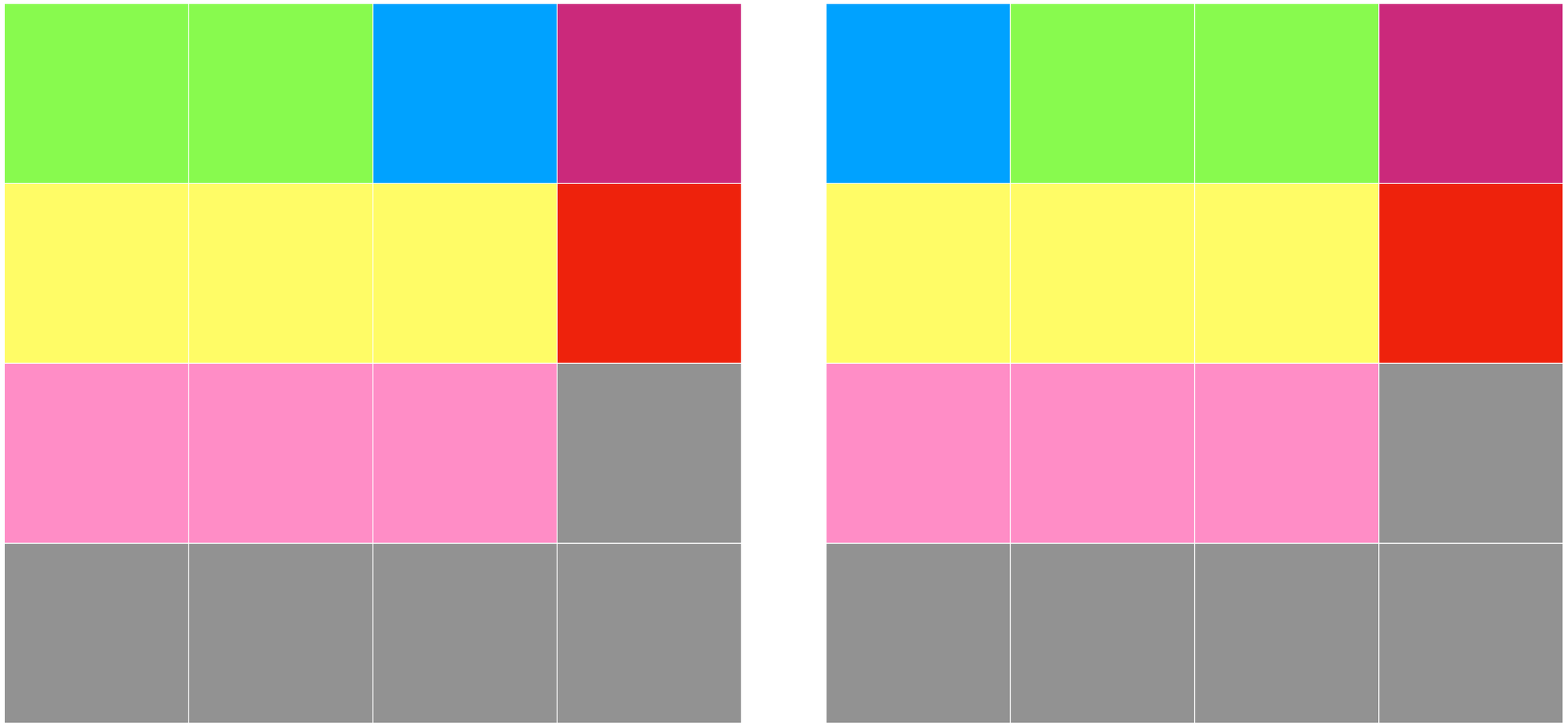




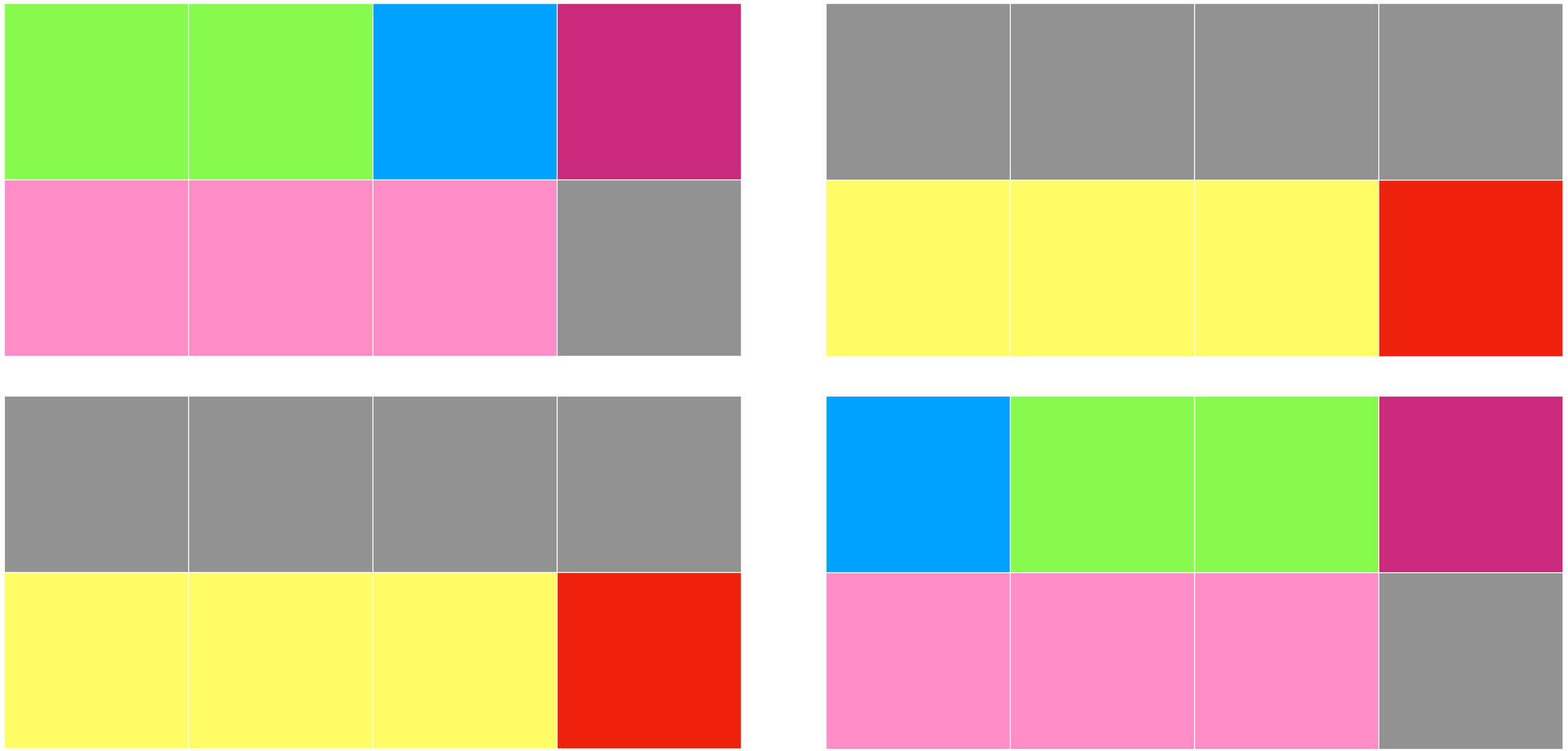
Both run the same deployments

- Let’s try to add some more
  - First: Two pods of 3 blocks each

Two nodes, preloaded



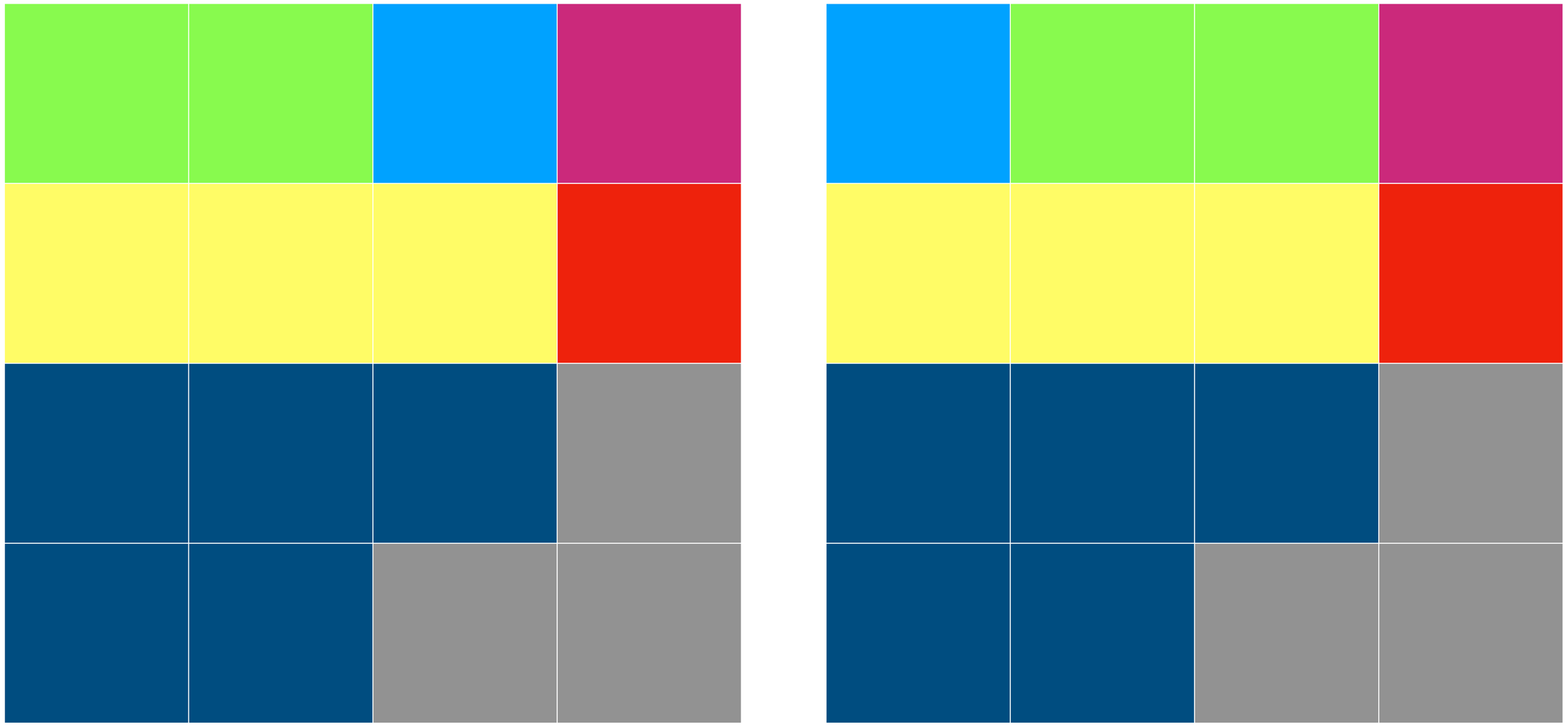
Four nodes, preloaded



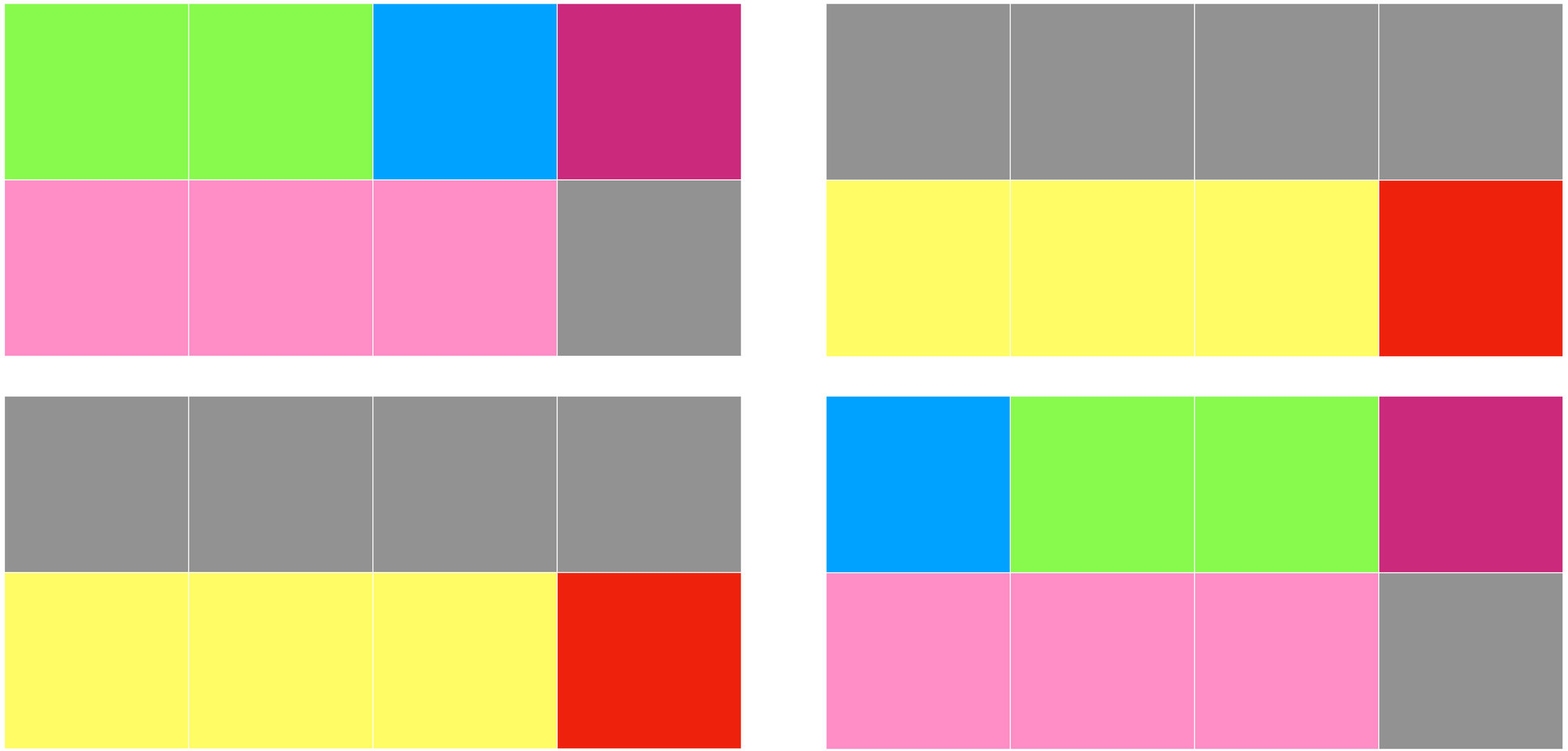
Both run the same deployments

- Let’s try to add some more
  - First: Two pods of 3 blocks each
  - And now a huge deployment: 5 blocks each

Two nodes, preloaded



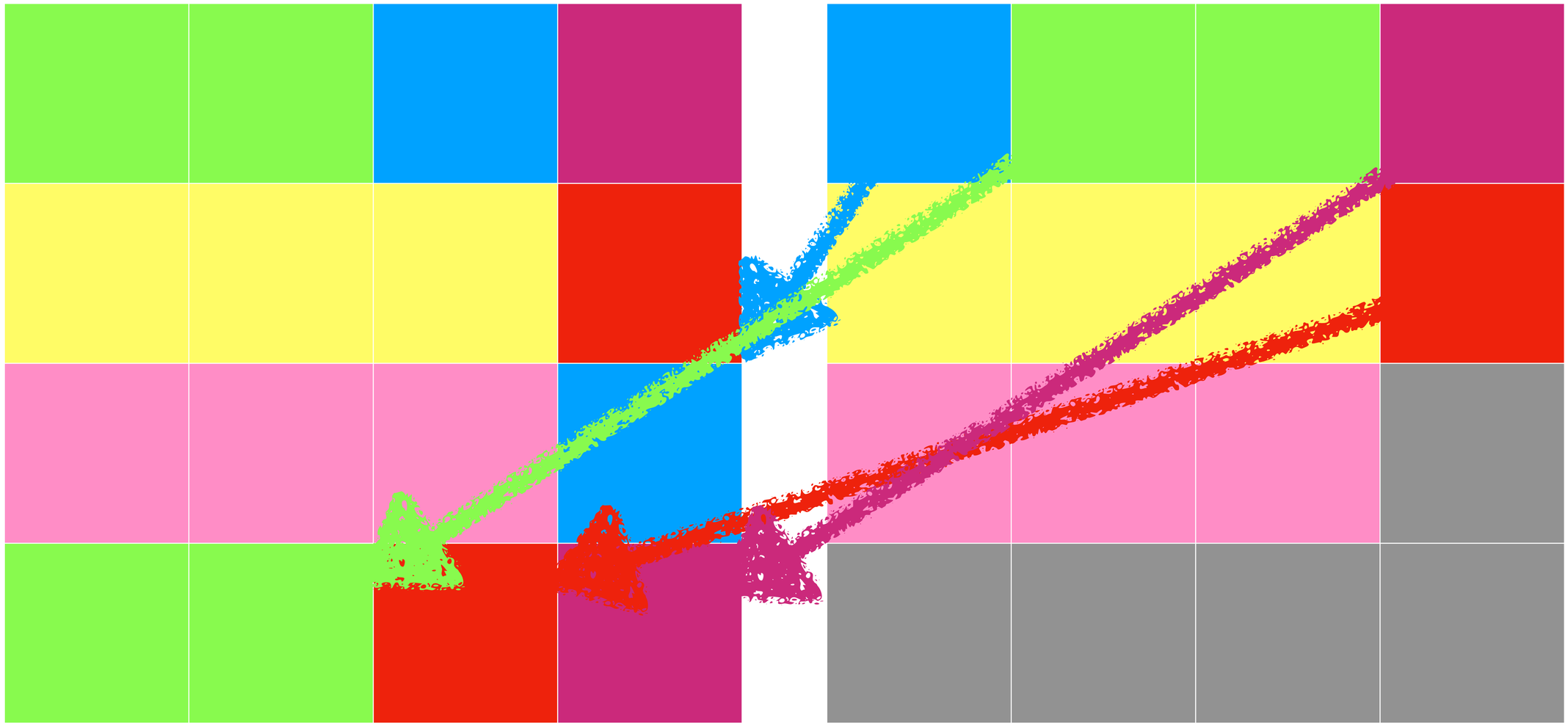
Four nodes, preloaded



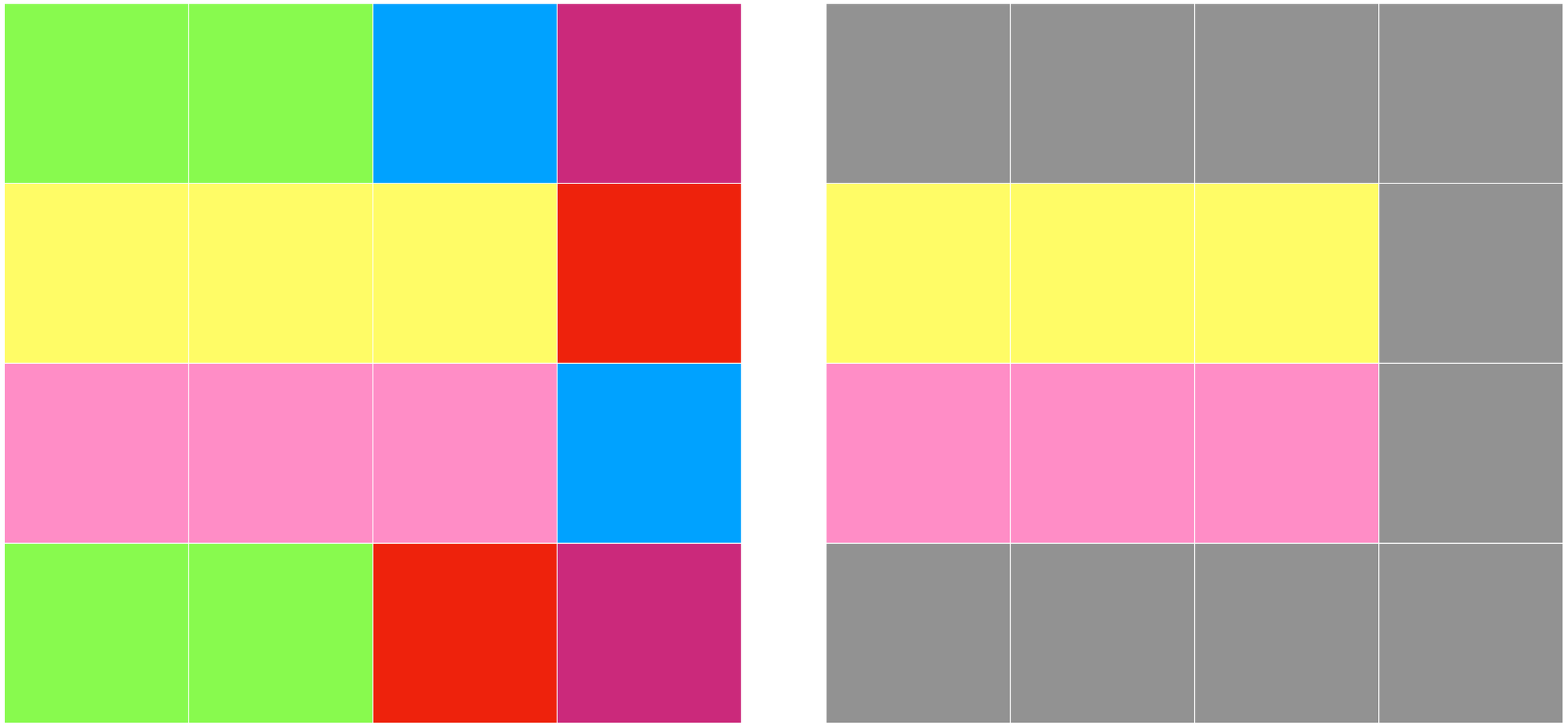
Both run the same deployments

- Let’s try to add some more
  - First: Two pods of 3 blocks each
  - And now a huge deployment: 5 blocks each
    We have a partial success….
  - Bigger is better for now!
- Let’s cordon a server for updates…

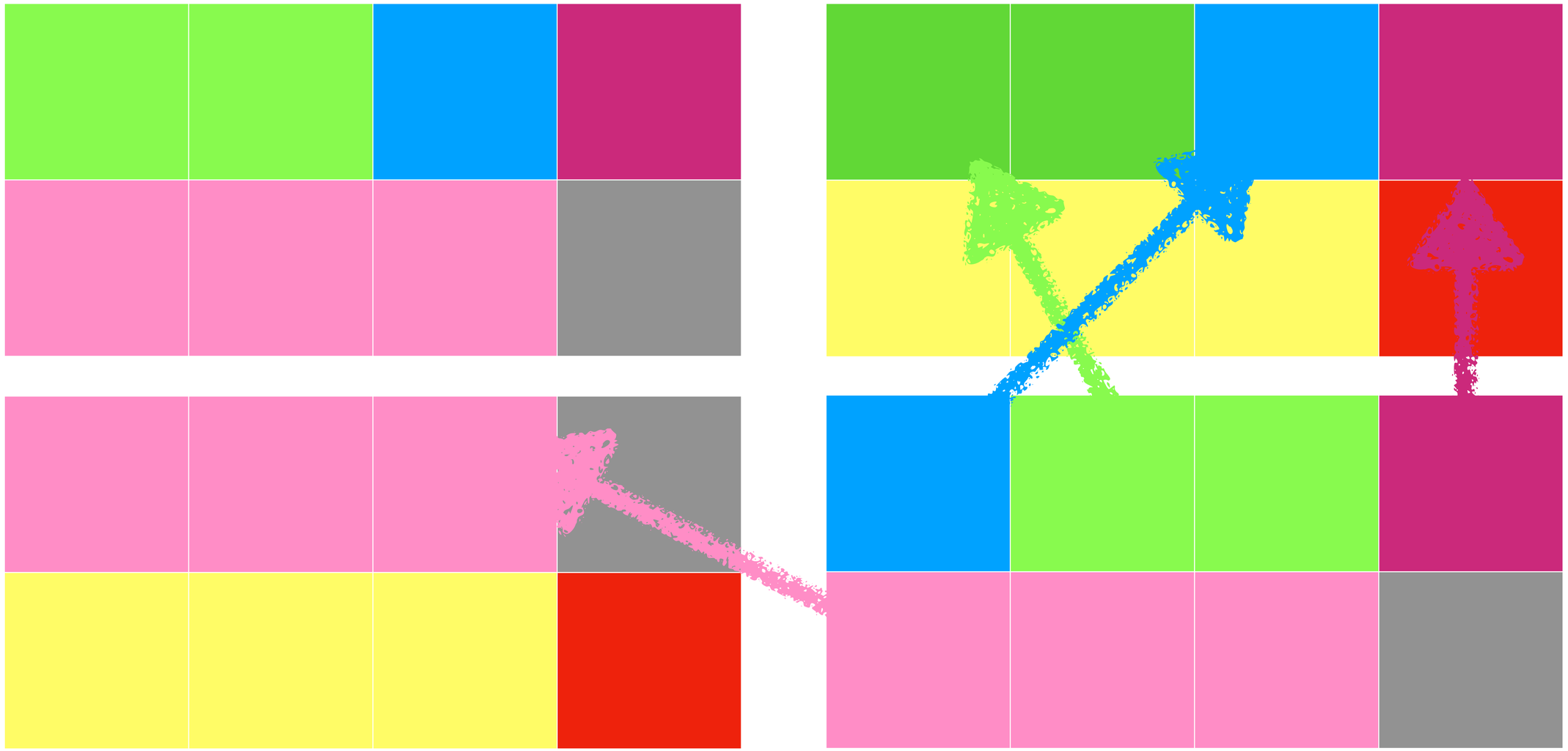
Two nodes, preloaded



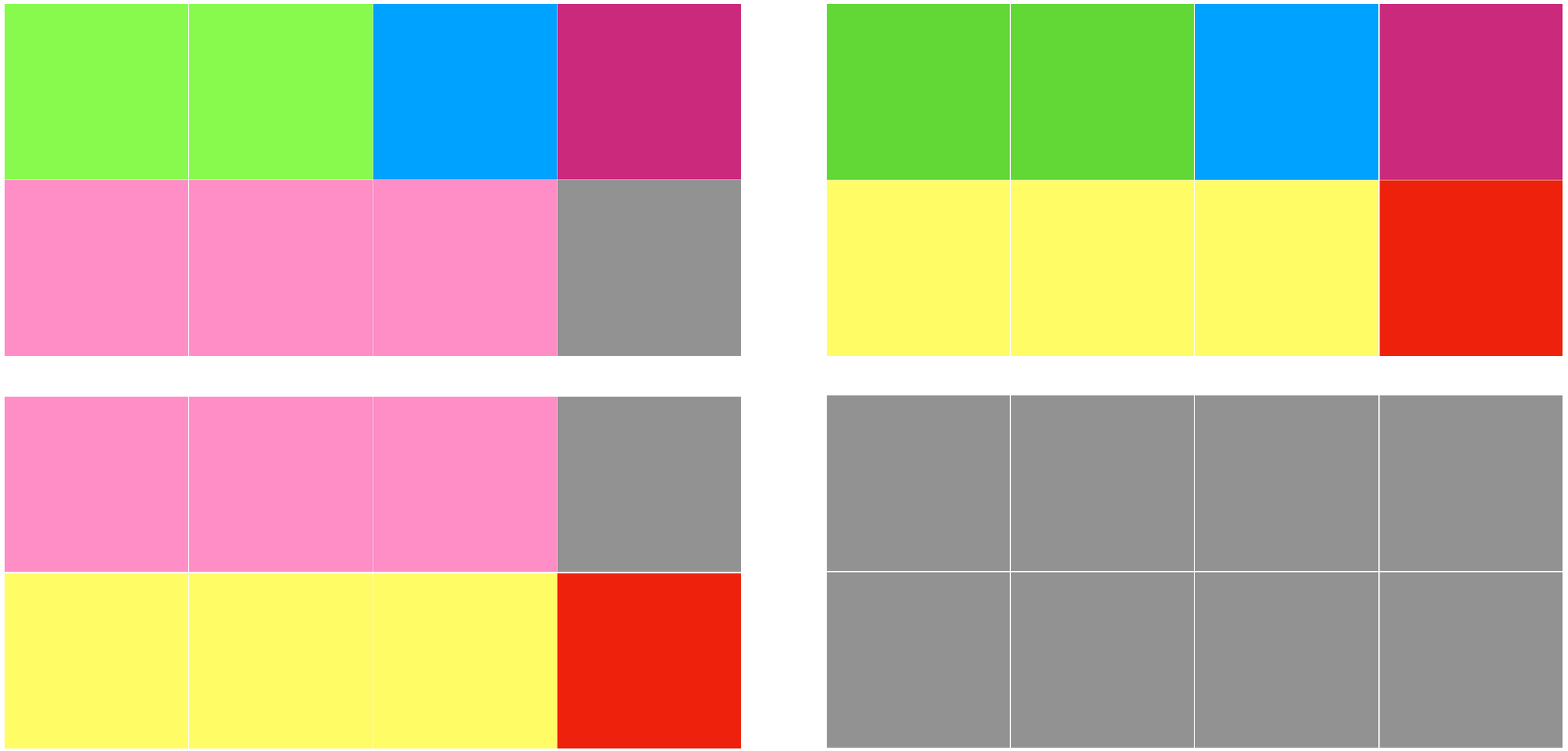
Two nodes, preloaded



Four nodes, preloaded



Four nodes, preloaded



Both run the same deployments

- Let’s try to add some more
  - First: Two pods of 3 blocks each
  - And now a huge deployment: 5 blocks each
    We have a partial success….
  - Bigger is better for now!
- Let’s cordon a server for updates…
  - And now smaller seems better for two reasons:
    1. All deployments are there
    2. No dual deployments om same node

Things to keep in mind

- When cordoning and draining a node all pods have to be redeployed somewhere else:
- What if:
  - Many images have to be downloaded by a single node?
  - The resource settings are off? (More later)
  - antiAffinity is set preventing pods to run on the same node as other pods from the same deployment?

Resources Request - Limits

- `requests`: The minimum amount of CPU and/or memory that has to be available on a a node before a container is deployed.
- `limits`: The maximum amount of CPU and/or memory that a container is allowed to use.
- In an ideal world these two are equal.
- If only `limits` are specified K8s will set `requests` to the same value (see above)
- If only `requests` are specified there are no limits… Really?
- https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#requests-and-limits

Demo-1

- Deploy Nginx pods who do not need memory but do `request` it
- Wat a waste…

Demo-2

```
...
resources:
  limits:
    memory: 200Mi
  requests:
    memory: 100Mi
...
```

- Let’s deploy a pod that `requests` 150M of memory… Next deploy a pod `requesting` more then 200M.
- Repeat the exercise with pods requesting only 10M, but scale 15 of them in parallel. After a while they’ll request 150M just as above.

Swap…

- Does not exist anymore. Paging is the successor of swap.
- But we call it swap….
- Swap can lead to Swap Death (Check Wikipedia)
- No paging (Swap!) on K8s nodes. Please? Please?? Please???
- https://github.com/kubernetes/kubernetes/issues/53533

Pressure

A hard eviction threshold has no grace period, and if observed, the kubelet will take immediate action to reclaim the associated starved resource. If a hard eviction threshold is met, the kubelet kills the Pod immediately with no graceful termination.

Node Condition	Threshold
MemoryPressure	memory.available<100Mi
DiskPressure	nodefs.available<10%
	nodefs.inodesFree<5%
	imagefs.available<15%

https://kubernetes.io/docs/tasks/administer-cluster/out-of-resource/

Reclaiming node level resources

- With imagefs (/var/lib/docker??):
  - If nodefs filesystem has met eviction thresholds, kubelet frees up disk space by deleting the dead Pods and their containers.
  - If imagefs filesystem has met eviction thresholds, kubelet frees up disk space by deleting all unused images.
- Without imagefs:
  - If nodefs filesystem has met eviction thresholds, kubelet frees up disk space in the following order:
  - Delete dead Pods and their containers
  - Delete all unused images

https://kubernetes.io/docs/tasks/administer-cluster/out-of-resource/

Probes

- Use probes to determine the health of pods:
  - Readyness signals an ingress controller and determines update speed
  - Liveness determines if a pod is restarted
- Several techniques can be used:
  - script
  - http request
  - TCP port

https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/

A probe script

```
#!/bin/bash

function probe_log() {

    echo "$( date +%D-%T ): ${1}" > /proc/1/fd/1

    exit 1

}

# Use wget and not curl... wget has easier exit codes!

wget  http://localhost:9876/health -o /dev/null || probe_log "Restarting because healt
page is not responding"

# Debug checks:

# Report a NOT ALIVE status if the file /tmp/notready does exist:

cat /tmp/notalive > /dev/null 2>&1 && probe_log "File /tmp/notalive exists"exit 0
```

In a configmap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: probes
  namespace: probes
data:
  readinessprobe.sh: |-
    #!/bin/bash
   ...
    exit 0
  livenessprobe.sh: |-
    #!/bin/bash
   ...
    exit 0
```

kubectl create configmap --from-file=/etc/hosts test --dry-run -o yaml

Mounted as a volume

```
  volumeMounts:
  - name: probes
    mountPath: "/probes"
    readOnly: true
volumes:
  - name: probes
    configMap:
      name: probes
```

Called from K8s

```
livenessProbe:
  periodSeconds: 1
  successThreshold: 1
  failureThreshold: 2
  initialDelaySeconds: 30
  timeoutSeconds: 5
  exec:
    command:
      - /bin/bash
      - /probes/livenessprobe.sh
readinessProbe:
  periodSeconds: 3
  successThreshold: 1
```

Demo

- Take it slowly... One step at a time.
- It is intentionally slow
- Time for Demo 3