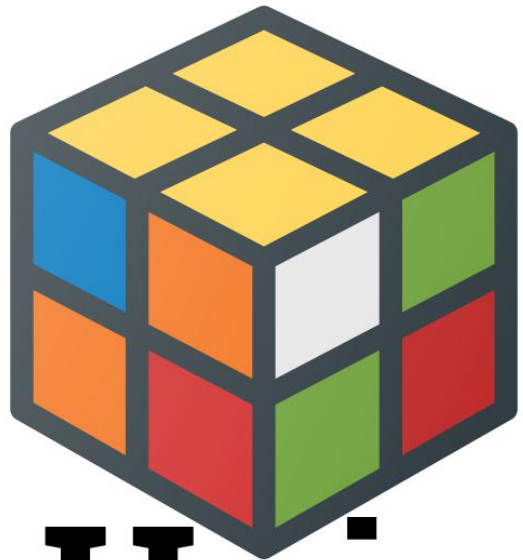


Geek University

Evolua seu lado geek!

www.geekuniversity.com.br

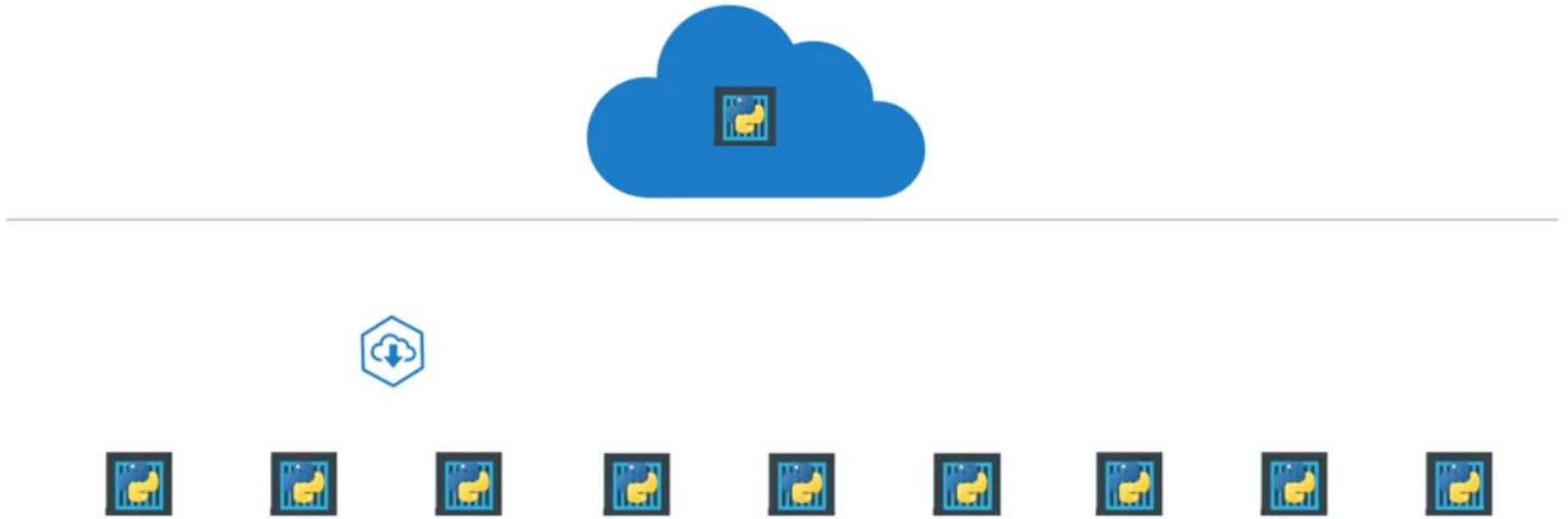
Conhecendo o Deployments

Conhecendo o Deployments

Imagine que tenhamos uma aplicação web desenvolvida em Python e precisamos realizar o deployment (publicar) nossa aplicação.

Conhecendo o Deployments

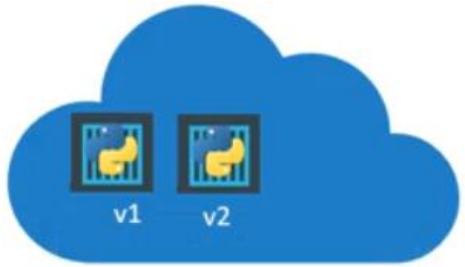
Então fazemos o deployment (a publicação) com vários containers para dividir a carga de trabalho da nossa aplicação que será um sucesso.



Conhecendo o Deployments

Depois de um tempo, implementamos novas funcionalidades e melhorias na nossa aplicação, e desta forma geramos uma nova versão para publicação.

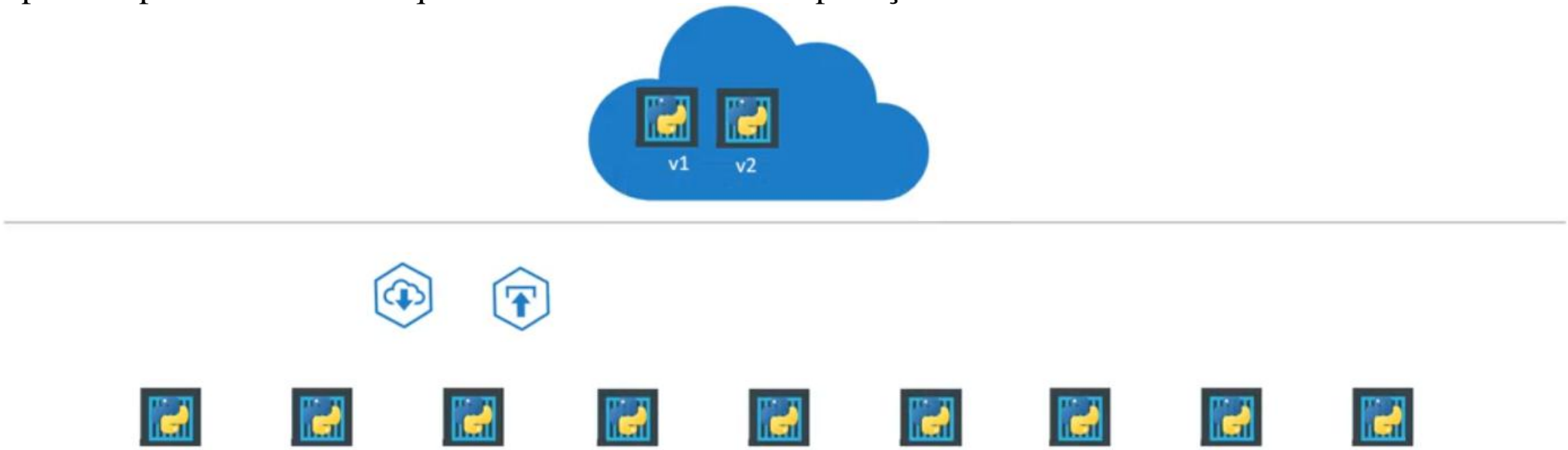
Como fazemos?



Conhecendo o Deployments

Geralmente podemos pensar que iremos simplesmente realizar a publicação da nova versão da nossa aplicação em todos os pods de uma vez só não é?

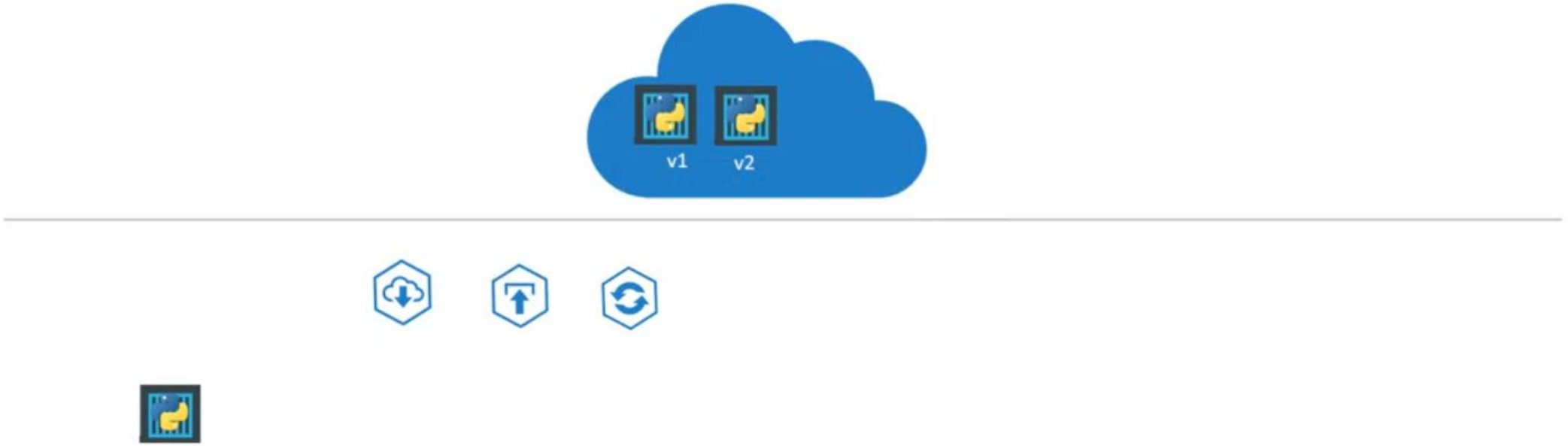
Mas isso pode impactar os usuários que estiverem acessando a aplicação. Não é o recomendado a se fazer.



Conhecendo o Deployments

O que queremos fazer é atualizar, um por um, nossos pods até que todos tenham sido atualizados.

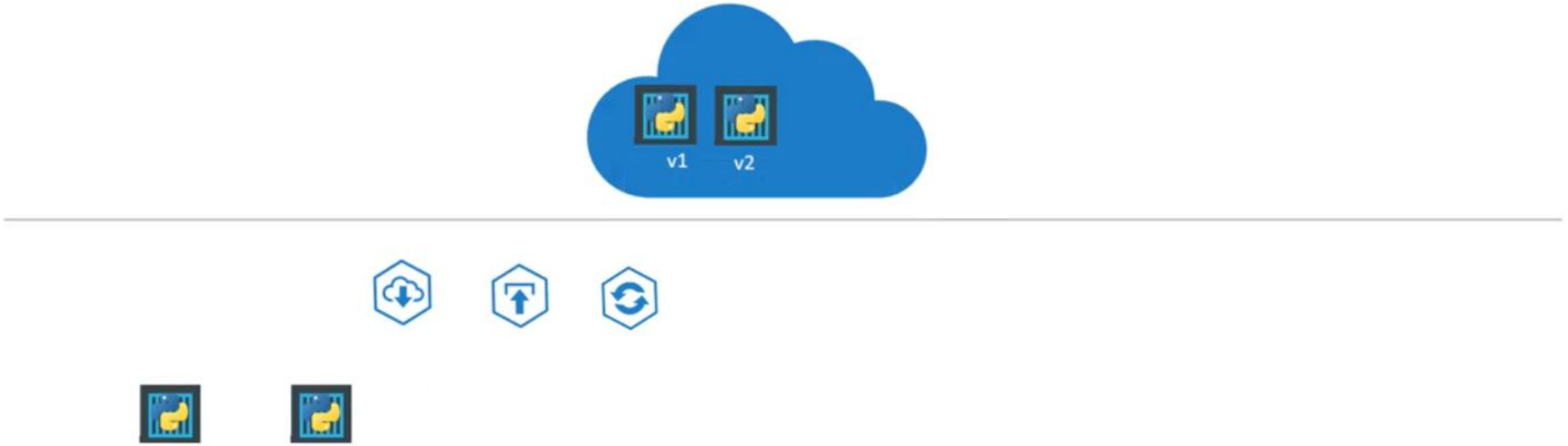
Isso é conhecido como “rolling release/rolling updates”



Conhecendo o Deployments

O que queremos fazer é atualizar, um por um, nossos pods até que todos tenham sido atualizados.

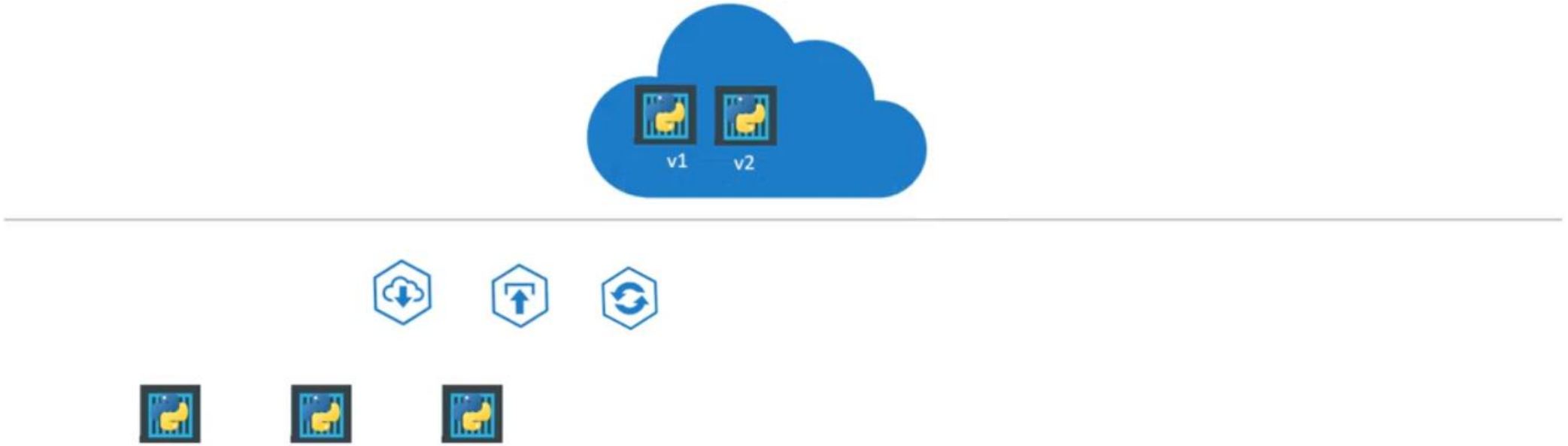
Isso é conhecido como “rolling release/roalling updates”



Conhecendo o Deployments

O que queremos fazer é atualizar, um por um, nossos pods até que todos tenham sido atualizados.

Isso é conhecido como “rolling release/roalling updates”



Conhecendo o Deployments

O que queremos fazer é atualizar, um por um, nossos pods até que todos tenham sido atualizados.

Isso é conhecido como “rolling release/roalling updates”

Conhecendo o Deployments

O que queremos fazer é atualizar, um por um, nossos pods até que todos tenham sido atualizados.

Isso é conhecido como “rolling release/roalling updates”

Conhecendo o Deployments

O que queremos fazer é atualizar, um por um, nossos pods até que todos tenham sido atualizados.

Isso é conhecido como “rolling release/roalling updates”

Conhecendo o Deployments

O que queremos fazer é atualizar, um por um, nossos pods até que todos tenham sido atualizados.

Isso é conhecido como “rolling release/roalling updates”

Conhecendo o Deployments

O que queremos fazer é atualizar, um por um, nossos pods até que todos tenham sido atualizados.

Isso é conhecido como “rolling release/roalling updates”

Conhecendo o Deployments

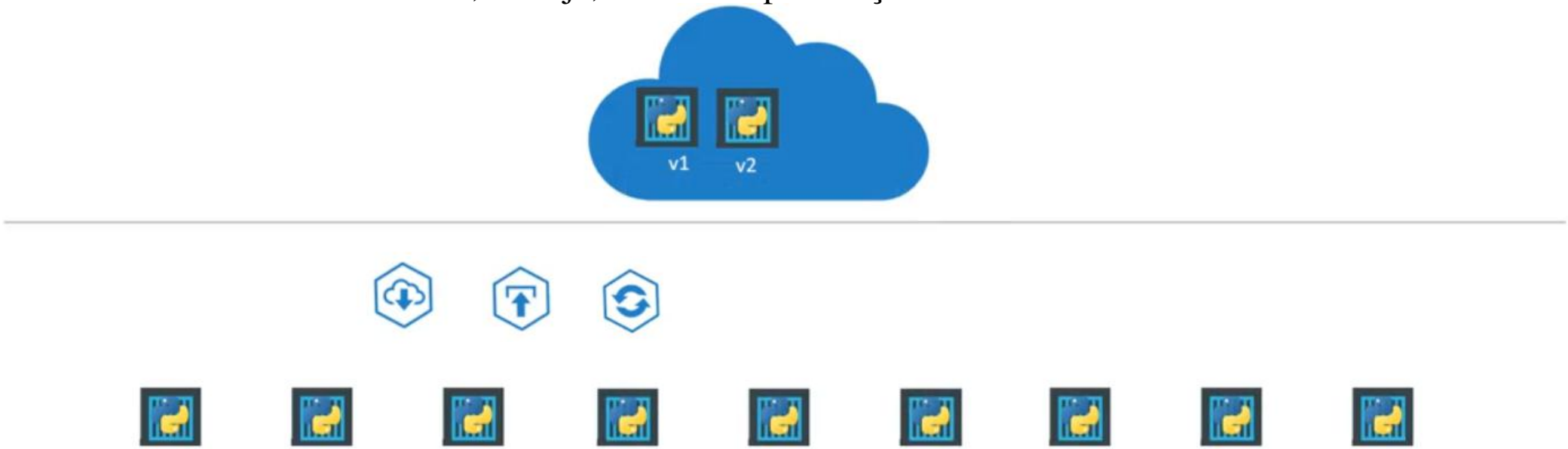
O que queremos fazer é atualizar, um por um, nossos pods até que todos tenham sido atualizados.

Isso é conhecido como “rolling release/roalling updates”

Conhecendo o Deployments

Suponha que um dos pods apresentou um problema inesperado durante a atualização, e você decida reverter a atualização para estudar o caso.

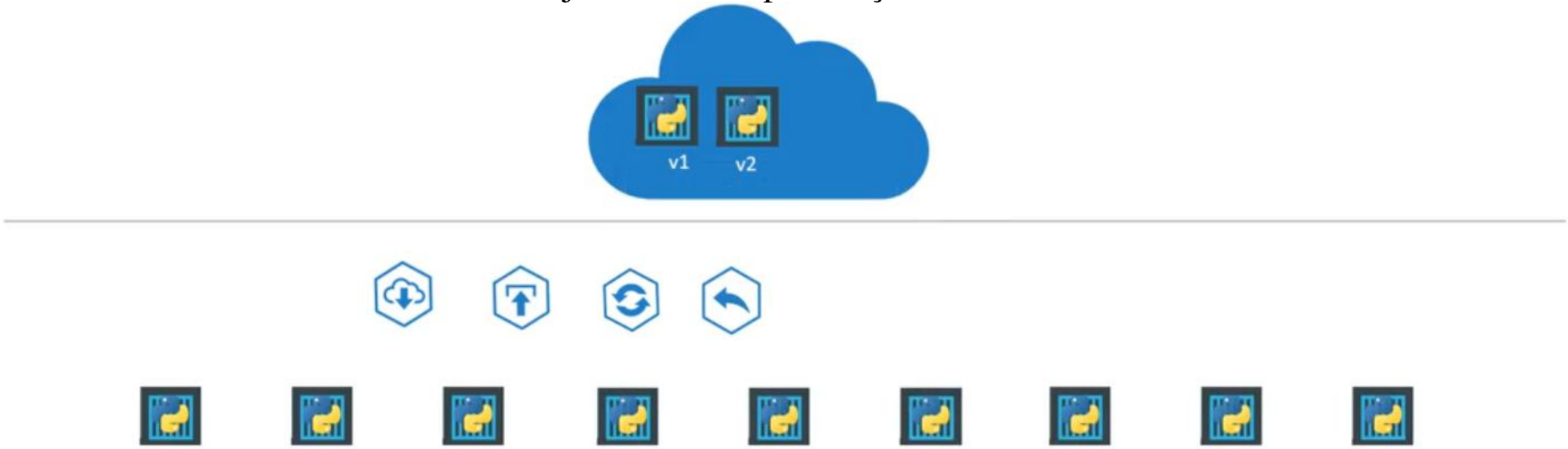
Então você deve executar um “rollback”, ou seja, desfazer a publicação.



Conhecendo o Deployments

Suponha que um dos pods apresentou um problema inesperado durante a atualização, e você decida reverter a atualização para estudar o caso.

Então você deve executar um “rollback”, ou seja, desfazer a publicação.



Conhecendo o Deployments

Suponha que um dia você resolveu fazer uma atualização mais complexa, não apenas publicando uma nova versão da aplicação web mas também altualizando a versão dos containers, ou do kubernetes, ou dos ambientes das aplicações nos containers e etc.

Conhecendo o Deployments

Você não deverá fazer isso com o ambiente em execução, mas sim pausar ou parar a execução, realizar as múltiplias atualizações e então reiniciar os serviços.

Esta é a chamada “manutenção programada do sistema”

Conhecendo o Deployments

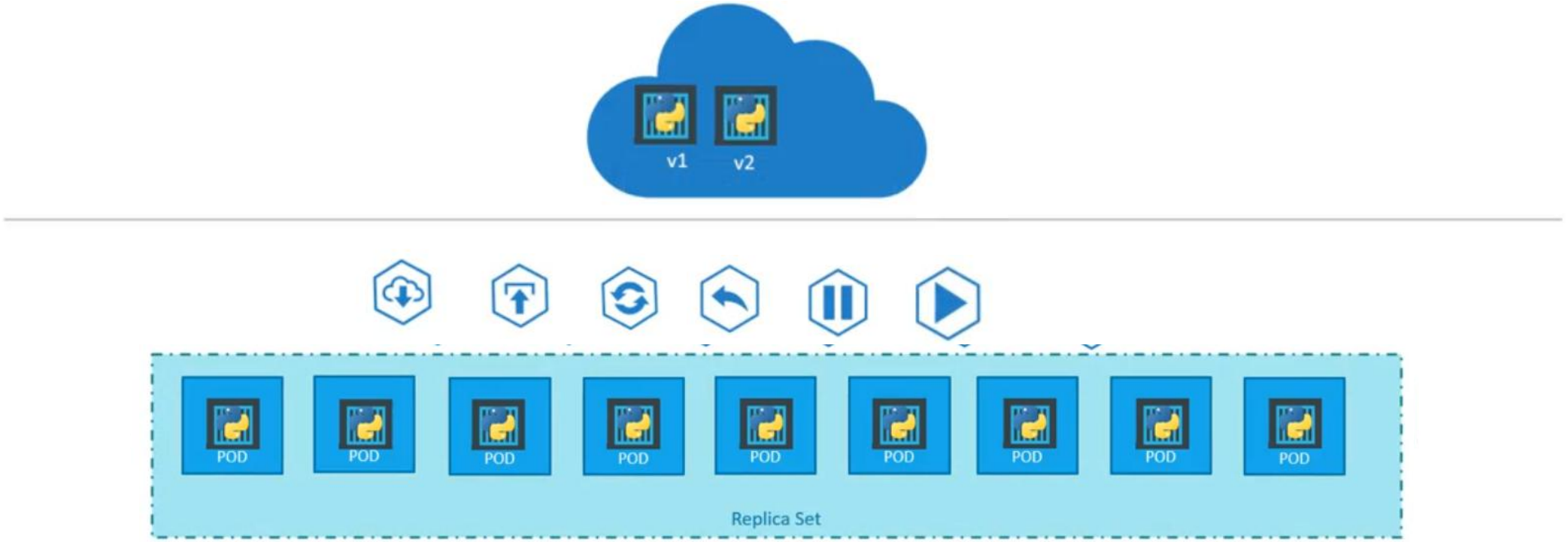
Tudo isso que discutimos até então, e um pouco mais, está disponível na ferramenta do Kubernetes chamada Deployment

Conhecendo o Deployments

Até então aprendemos que nossos containers estão encapsulados em pods e fazem parte de um cluster Kubernetes.

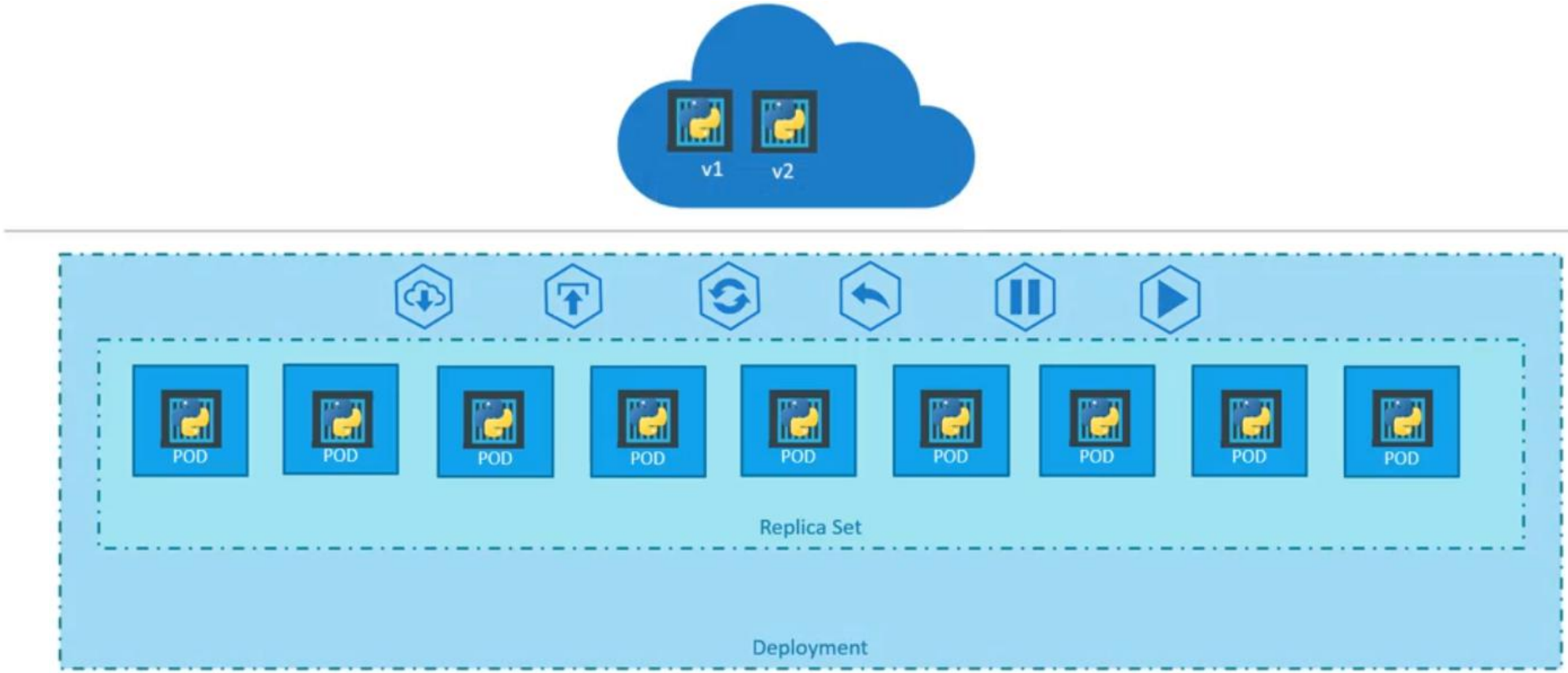
Conhecendo o Deployments

Aprendemos a adicionar alta disponibilidade à nossa aplicação fazendo uso do Replica Set



Conhecendo o Deployments

Chegamos então ao Deployment, que “encapsula” todo este processo adicionando todos os recursos discutidos até aqui.



Conhecendo o Deployments

Definição do Deployment em YAML

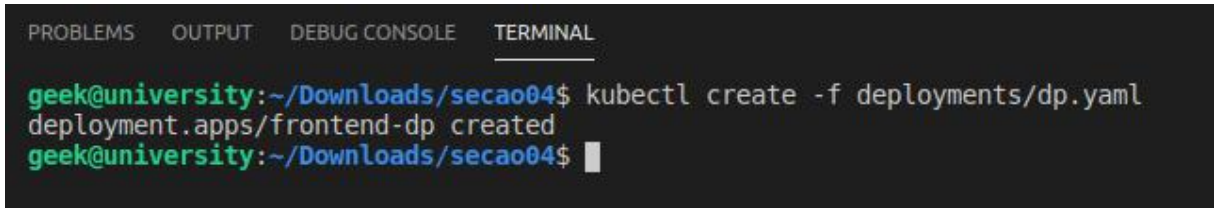
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-dp
  labels:
    app: frontend-app
    type: frontend
spec:
  template:
    metadata:
      name: frontend-pod
      labels:
        app: frontend-app
        type: frontend
    spec:
      containers:
        - name: nginx-container
          image: nginx
  selector:
    matchLabels:
      type: frontend
  replicas: 3
```

O arquivo de definição do deployment é exatamente igual ao do Replica Set, diferenciamento apenas o tipo do objeto, neste caso Deployment.

Conhecendo o Deployments

Kubectl

A criação do objeto deployment não podia ser diferente dos outros objetos que já estudamos, pois tudo segue um padrão e uma lógica na computação:



Conhecendo o Deployments

Kubectl

A mesma lógica se aplica para consultarmos os deployments existentes no nosso cluster:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

geek@university:~/Downloads/secao04$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
frontend-dp   3/3     3            3           46s
geek@university:~/Downloads/secao04$
```

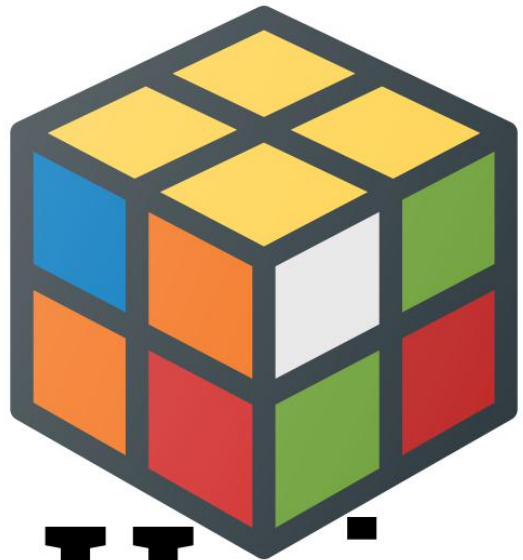
Assim como para consultar os pods:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

geek@university:~/Downloads/secao04$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
frontend-dp-79fdffd664-ftjv2   1/1     Running   0          98s
frontend-dp-79fdffd664-gq5wz   1/1     Running   0          98s
frontend-dp-79fdffd664-jkpjv   1/1     Running   0          98s
geek@university:~/Downloads/secao04$
```

Ou mesmo replicaset que foi criado pelo próprio deployment:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

geek@university:~/Downloads/secao04$ kubectl get replicaset
NAME                     DESIRED   CURRENT   READY   AGE
frontend-dp-79fdffd664   3         3         3       18m
geek@university:~/Downloads/secao04$
```

OBS: Na próxima aula iremos realizar o exercício prático.



Geek University

Evolua seu lado geek!

www.geekuniversity.com.br