Engineering a Kubernetes Operator

Lessons Learned Versions 1 to 5

Andrew L’Ecuyer
Sr. Director of Kubernetes Engineering
crunchy data

About Me

- Sr. Director of Kubernetes Engineering at Crunchy Data
- Manage the Engineering Team responsible for PGO, Crunchy Data's Postgres Operator
- I have experienced firsthand the **explosion of operators in recent years**

Email: andrew.lecuyer@crunchydata.com
Discord: andrewlecuyer
GitHub: andrewlecuyer

crunchydata

Crunchy Data: Postgres Anywhere

BARE METAL, VMs, CLOUD

Crunchy Postgres

Crunchy Certified PostgreSQL is production ready Postgres.

INCLUDES:

- ✓ Backups
- ✓ Disaster recovery
- ✓ High availability
- ✓ Monitoring
- ✓ Automation
- ✓ Self managed

KUBERNETES

Crunchy Postgres for Kubernetes

Cloud Native Postgres on Kubernetes powered by Crunchy Postgres Operator.

INCLUDES:

- ✓ Simple provisioning
- ✓ Backups and DR included
- ✓ High availability
- ✓ Seamless upgrades
- ✓ Scale from 1 to thousands of databases
- ✓ Self managed

FULLY MANAGED CLOUD

Crunchy Bridge

The fully managed Postgres option on your choice of Cloud provider.

INCLUDES:

- ✓ AWS, Azure or GCP
- ✓ Continuous protection
- ✓ Backups
- ✓ Point in Time Recovery
- ✓ Pay for what you use
- ✓ The developer experience you want

crunchy data

Outline

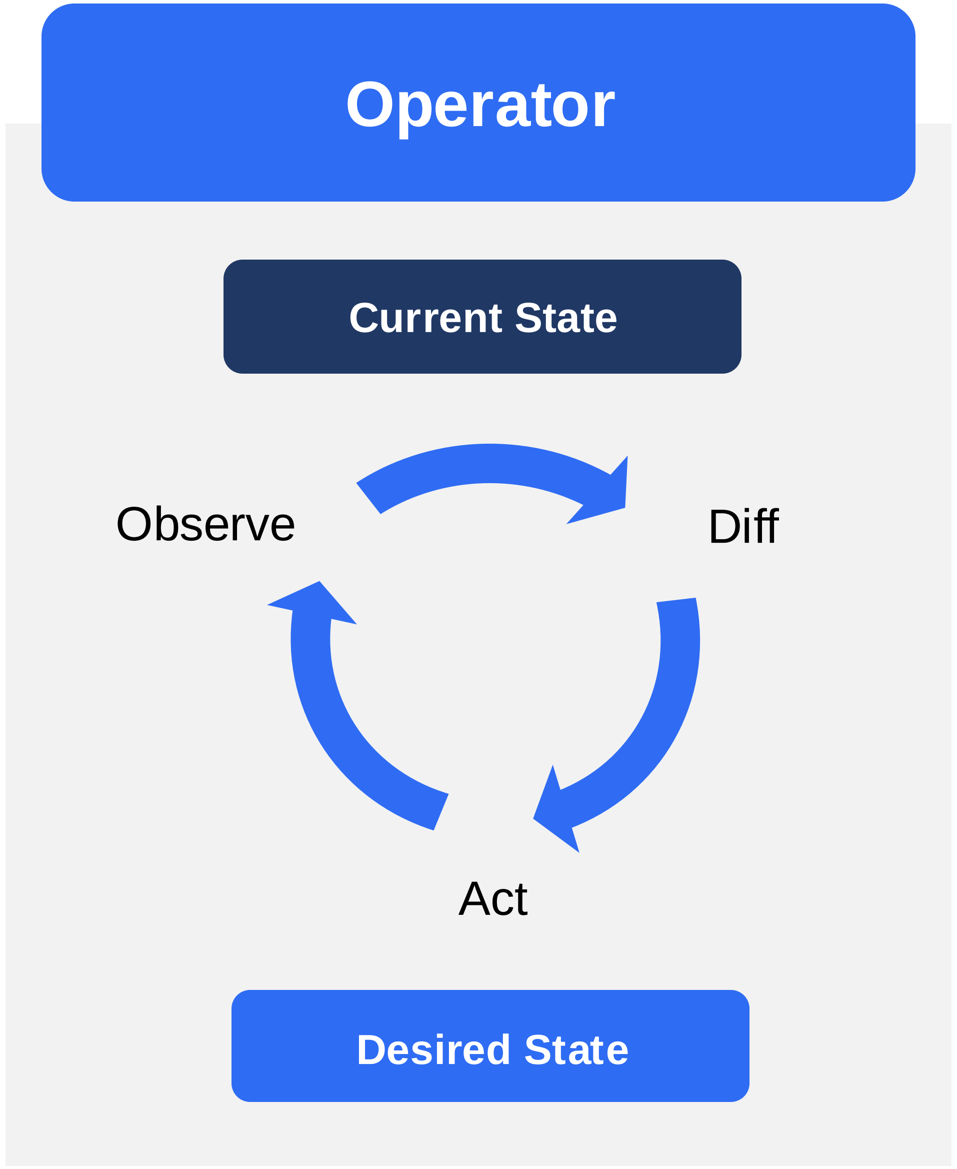
- Insights & lessons-learned from Crunchy Data’s journey building the first five versions of a Kubernetes Operator for Postgres
- Will specifically focus on: High Availability (HA), Disaster Recovery (DR) and Upgrades
- Will highlight important evolutions within Kubernetes (and Postgres) that have empowered operator development in recent years
- As you will see, there has never been a better time to build an operator!

crunchydata

Kubernetes Operator

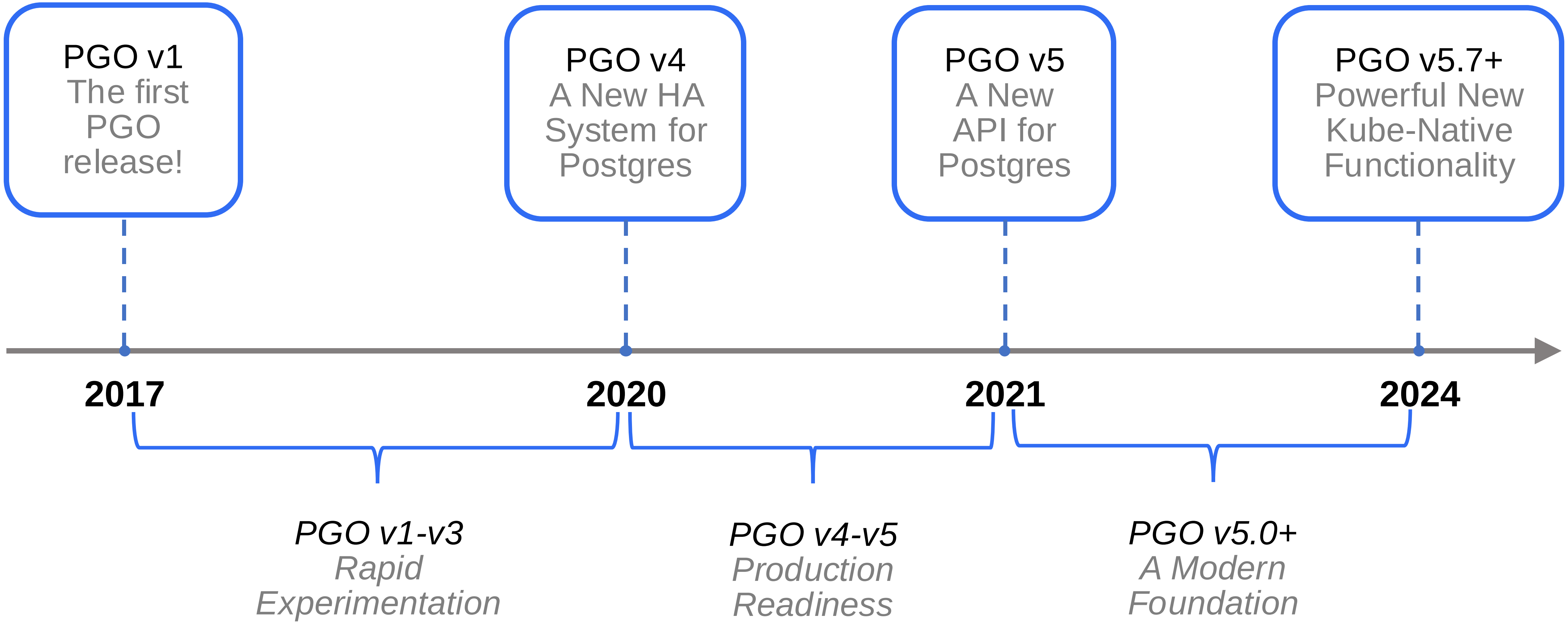
Operators manage complexity. Adding HA, DR and seamless upgrades to Postgres isn't easy!

Operators bring new user communities into Kubernetes by making Kubernetes accessible



crunchydata

PGO, the Postgres Operator from Crunchy Data



Kubernetes Landscape in 2017

2014

June 7 – Initial release

July 21 – Kubernetes v.10 gets released

Feb 23 – First release of Helm

July 21 – Kubernetes v1.0 is released and CNCF is established

Nov 3 – OpenShift joins the project

Sep 29 – Pokémon GO! Case study

Nov 16 – Helm 2.0.0 is released

March 27 – PGO 1.0 released

March 28 – Kubernetes v1.6 release moves dynamic storage provisioning to stable

2024

March 27, 2017
PGO 1.0 released

March 28, 2017
Kubernetes v1.6 released

Kubernetes Tooling

- Helm still on version 2
- Kustomize has not been released

Kubernetes API

- The new StatefulSets API is in beta, after being renamed from the "PetSet" API in Kubernetes v1.5
- StorageClass and dynamic volume provisioning were promoted to stable in Kubernetes v1.6

Operator Tooling

- Kubebuilder, Operator SDK and Controller Runtime projects do not exist
- Primary focus is getting applications and services up and running

kubebuilder OPERATOR FRAMEWORK Kustomize.io HELM

High Availability

crunchy data

High Availability: Design Considerations

Use an existing HA solution for Postgres that is Kubernetes-ready?

Vs.

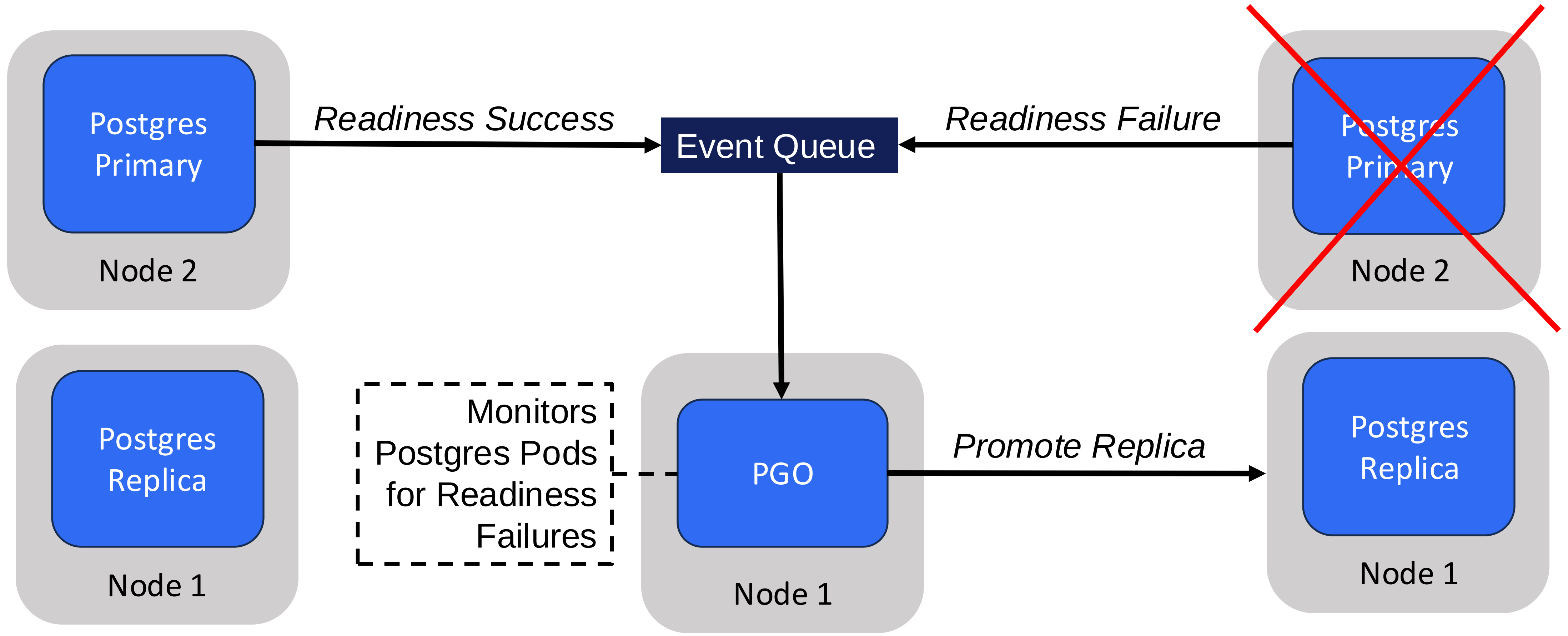
Or build a custom HA solution for Postgres using the operator?

Ideally both Postgres and the operator should be highly-available

...

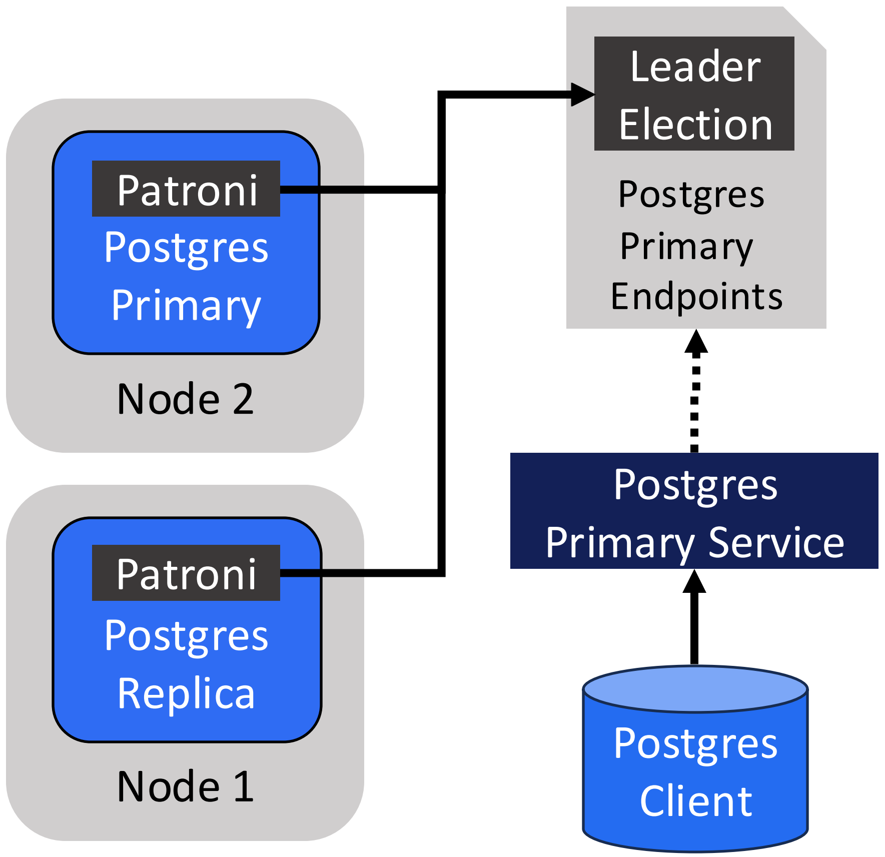
However, database availability is the top priority!

Postgres High Availability: PGO Versions 1-3

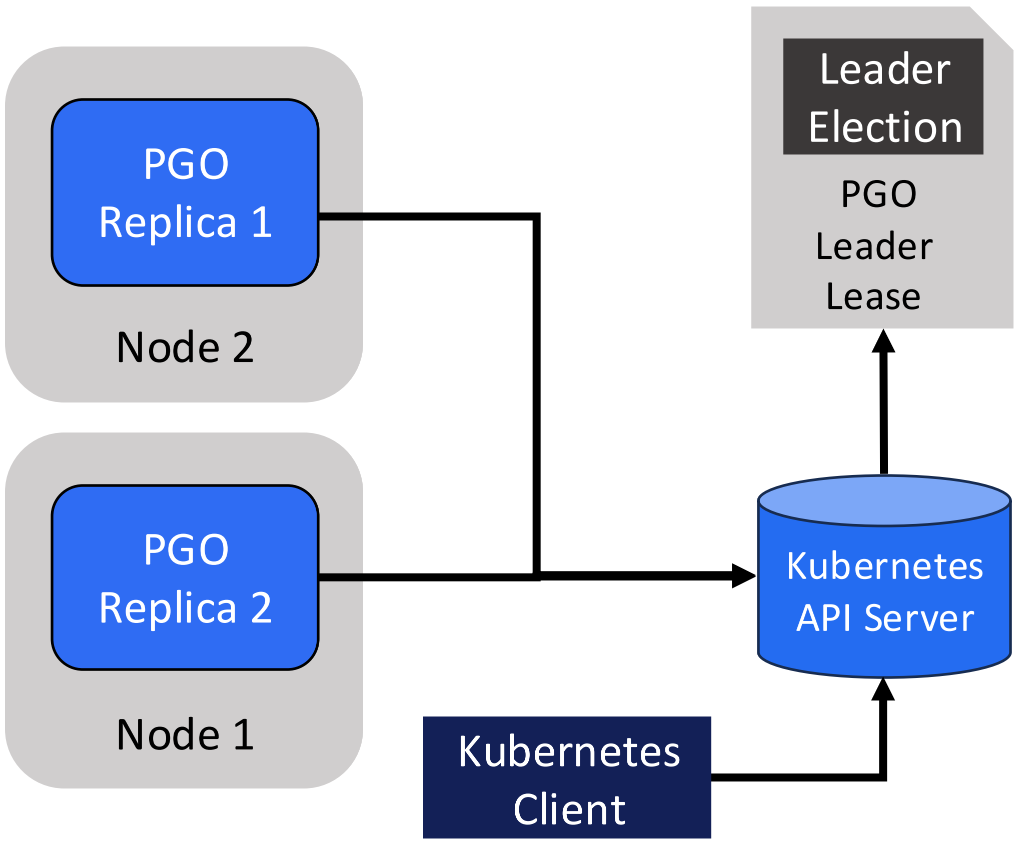


High Availability: Current Solution

Postgres



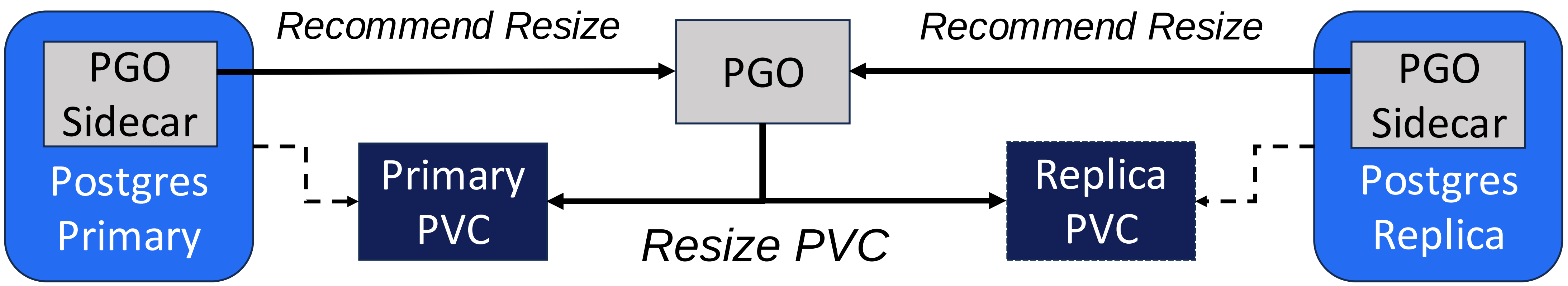
PGO



> Postgres High Availabilty (HA) demo here

> Postgres Operator High Availabilty (HA) demo here

Auto-Grow Evolution



Run a PGO sidecar in each Postgres instance Pod to determine when we're running out of storage space. Then, use Kubernetes PVC volume expansion to allow automatic resizing of PVC's without a rolling update

crunchy data

Upgrades

crunchy data

Upgrades: Design Considerations

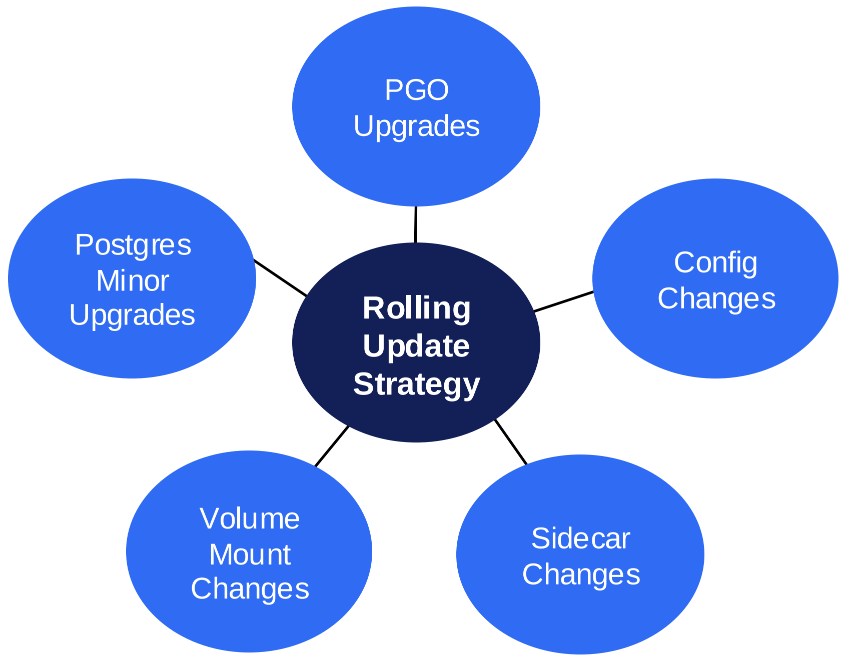
Fully automate any/all upgrades?

Vs.

Or require manual intervention in certain places?

Upgrades: Solution

A safe, fully-automated rolling update strategy for most upgrades & changes



A user-initiated, semi-automated strategy for Postgres major version upgrades

Steps:

1. Take a full backup
2. Create a PGUpgrade resource
3. Shutdown & annotate the cluster
4. Wait for upgrade to complete
5. Start the cluster
6. Complete post-upgrade tasks

> Rolling update demo here

```
> Postgres major version upgrade demo here
```

Disaster Recovery

crunchy data

Disaster Recovery: Design Considerations

Build a custom solution on top of existing DR tooling?

Vs.

Or work with the Postgres community to better align DR tooling with Kubernetes?

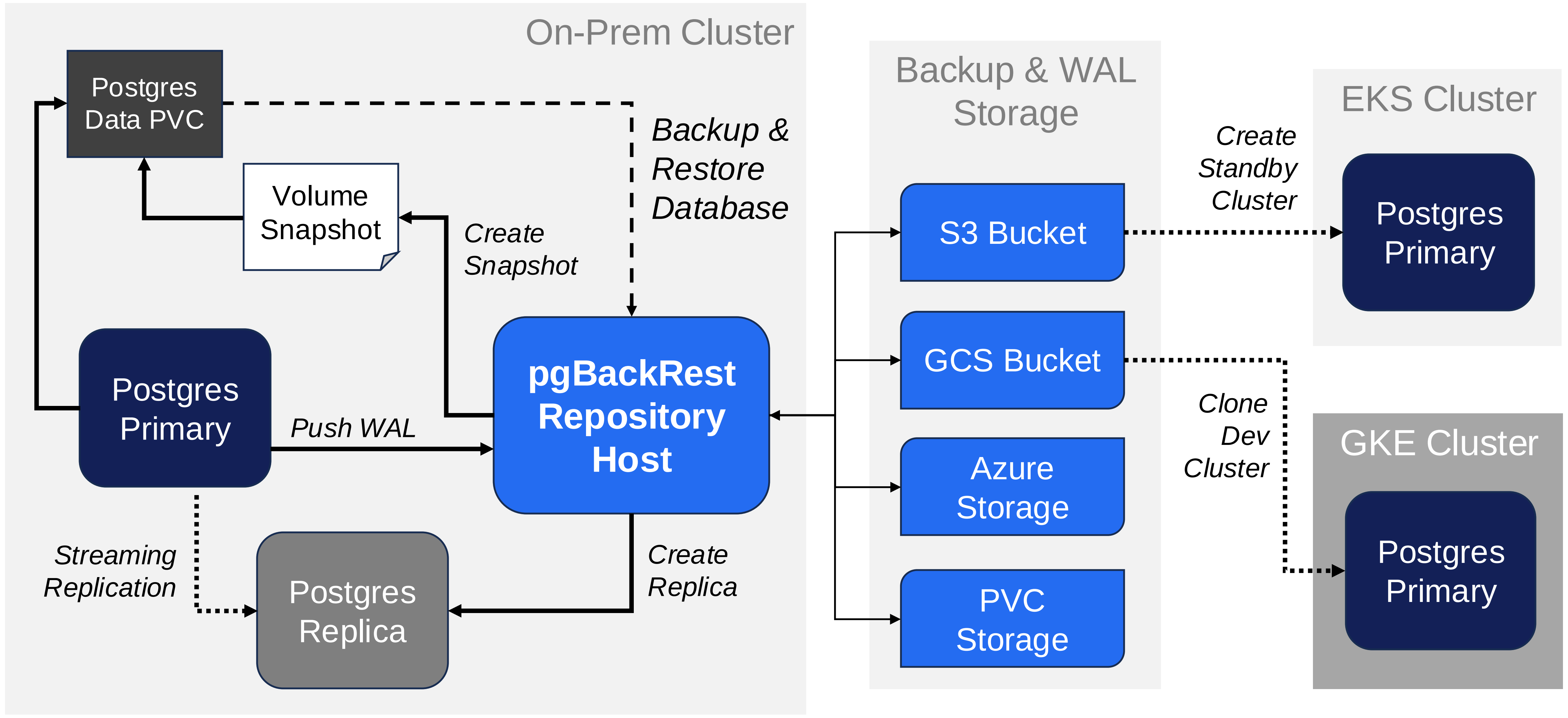
Stick with Postgres-native solutions for Disaster Recovery?

Vs.

Or use Kubernetes-native solutions for DR?

crunchy data

Disaster Recovery: Solution



crunchy data

> Disaster Recovery (DR) demo here

Summary

crunchy data

HA: Solution & Lessons Learned

PGO Solution Summary: Patroni for Postgres High Availability, controller-runtime for operator/PGO High Availability and Kubernetes volume expansion for auto-growing disks.

- Fight the "Not Invented Here" syndrome, and embrace existing solutions within the community
- A decentralized architecture allows us to scale
- Prevention is better than preparedness

Upgrades: Solution & Lessons Learned

PGO Solution Summary: A safe rolling update strategy for config changes, minor Postgres upgrades and PGO upgrades, and an orchestratable solution for Postgres major version upgrades

- Manage risk associated with upgrade automation, and only automate when risks can be mitigated
- When we can’t automate, ensure we can orchestrate
- Use status, conditions & events for upgrade visibility, and to empower engineers to safely perform upgrades

DR: Solution & Lessons Learned

PGO Solution Summary: pgBackRest for multi-cloud backup/restore functionality & data mobility, and Volume Snapshots to improve restore performance

- Focus on recovery rather than backups
- A robust DR solution can enable data mobility
- Use Postgres-native solutions to *safely* utilize Kubernetes-native solutions

Conclusion

crunchy data

Should You Build an Operator?

- **A great solution for Postgres!** Manages the complexity of deploying & managing a production-ready Postgres cluster
- More knowledge is available than ever before (documentation, blogs, books, etc.) for operator development
- Multiple mature operator frameworks to help you get started
- **Provides practical knowledge & skills for contributing back to Kubernetes**

Kubernetes Landscape in 2024

2014

June 7 – Initial release

July 21 – Kubernetes v.10 gets released

Feb 23 – First release of Helm

July 21 – Kubernetes v1.0 is released and CNCF is established

Nov 3 – OpenShift joins the project

Sep 29 – Pokémon GO! Case study

16 – Helm 2.0.0 is released

March 27 – PGO 1.0 released

March 28 – Kubernetes v1.6 release moves dynamic storage provisioning to stable

August 13, 2024
Kubernetes v1.31 released

March 18, 2024
PGO v5.7 released

2024

Kubernetes Tooling

- Helm now on version 3
- Kustomize now included in kubectl

Kubernetes API

- Stateful deployments are first-class within the Kubernetes platform, with a StatefulSet API that is both stable & mature

Operator Tooling

- Kubebuilder, Operator SDK & controller-runtime projects stable & mature
- Focus on advanced needs, e.g. multi-cluster, security, supply chain, and more

Thank You & Happy Building!

Come find me at the Crunchy Data booth to continue the conversation!

I look forward to hearing about your operator experiences.