

Staging Environment Blog:

Scheduled 7/26 @ 6:00 AM PDT

Post 1:

What do Hamilton and #SoftwareDevelopment have in common? It's all about the staging.

Both #developers and operators need the right tools and to follow some simple best practices to deliver good code fast. In this article, we break down four important considerations for staging environments: <https://www.architect.io/blog/2022-04-20/what-is-staging-environment/>

Scheduled 8/29 @ 6:00 AM PDT

Post 2:

Building and maintaining the infrastructure for one production-grade environment is hard. So, what if we stopped thinking about the staging environment as a second production environment, with duplicative costs and responsibilities?

<https://www.architect.io/blog/2022-04-20/what-is-staging-environment/> #DevOps

Orchestration vs choreography:

Scheduled 11/10

LinkedIn Post 2:

Like a musical orchestra, #microservice orchestration enables the execution of a query or process by allowing each microservice to implement tasks assigned by the orchestrator (conductor).

As opposed to the orchestration workflow, microservice choreography follows a decentralized mechanism. In choreography, the orchestrator does not exist. Instead, the microservices follow an asynchronous approach that entails executing queries when an event is raised.

So, which method should you choose?

<https://www.architect.io/blog/2022-06-30/microservices-orchestration-primer/>

Scheduled 9/13

Twitter Post 1:

You don't have to be a musical expert to understand the differences between #microservices orchestration and choreography. Let us show you the steps:

<https://www.architect.io/blog/2022-06-30/microservices-orchestration-primer/>

#SoftwareDevelopment

Scheduled 12/12

LinkedIn Post 3:

Orchestration or choreography? Choosing the right communication pattern for your microservices depends on the task at hand, the scale of the organization, and a variety of other factors.

If your application or software requires constant updates and new features ➡ Choreography

When there are complex workflows with thousands of microservices requiring levels of coordination and sync ➡ Orchestration.

Or you could even adopt a hybrid system.

<https://www.architect.io/blog/2022-06-30/microservices-orchestration-primer/>

6 CI/CD Best Practices:

Scheduled 9/20

LinkedIn Post 1:

Continuous integration refers to having developers continuously merge changes to a shared main branch, and continuous delivery allows you to automatically merge successful pull requests into that shared main branch.

CI/CD not only helps deliver faster but, when done well, also helps improve the overall quality and security of the software.

In this post, we outline the 6 CI/CD best practices you should follow:

<https://www.architect.io/blog/2022-07-13/six-cicd-best-practices/>

#SoftwareDevelopment #DevOps

LinkedIn Post 2

Scheduled 11/21

Twitter Post 1

Pro tip: Don't treat your CI/CD as only deployment automation.

CI/CD should be about your whole #SoftwareDevelopment life cycle, which means testing the code and preventing the deployment of buggy software:

<https://www.architect.io/blog/2022-07-13/six-cicd-best-practices/>

LinkedIn Post 3

Scheduled 12/14

Twitter Post 2

What do automated testing and rollback have in common? They're 2 of the 6 CI/CD best practices you need to know. What do you think the other 4 are?

Find out: <https://www.architect.io/blog/2022-07-13/six-cicd-best-practices/>

Scheduled 10/18

LinkedIn Post 4:

If your CI/CD pipeline is slow or difficult to use, your #developers will try to cut corners and bypass the pipeline whenever possible. Make sure that your CI/CD server or service is as responsive and simple as possible:

<https://www.architect.io/blog/2022-07-13/six-cicd-best-practices/>

#DevOps #SoftwareDevelopment

Scheduled 7/28 @ 6:00 am PDT

-> Moved until later because another post was scheduled that day

Dynamic Credentialing:

Allowing apps to be deployed without manual credential generation is critical to making item portable and deployable by anyone. In this tutorial, we describe what dynamic credentialing is and apply it to a practical use-case – integration with a hosted service, Auth0:

<https://www.architect.io/blog/2020-06-04/how-dynamic-credentialing-makes-apps-portable/>

#APIDevelopment #SoftwareDevelopment

Dependency Management

Scheduled 7/15 @ 6:00 am PDT

Post 1:

There's no doubt that #microservices have introduced an enormous amount of complexity into the design and operation of cloud software. How can #developers take advantage of the benefits of service-oriented design without introducing friction and cost elsewhere?

We've got the answer:

<https://www.architect.io/blog/2020-09-16/why-distributed-apps-need-dependency-management/>

Scheduled 8/10 @ 6:00 am PDT

Post 2:

It's time to unlock the power of distributed software and the 🔑 has been around for decades. #Developers have been using dependency management to collaborate and contribute to webs of interconnected components.

Here are a few of the reasons why it'll help us power the future of cloud-native development:
<https://www.architect.io/blog/2020-09-16/why-distributed-apps-need-dependency-management/>

#DevOps

Developer's Guide to GitHubs

Schedule 7/18 @ 6:00 AM PDT

Post 1:

How do #developers deploy features as quickly and safely as possible? It requires constant stability, and automation helps with both speed and stability. Now let's hook that automation tactic into a critical tool that already exists in your developers' workflow, Git:

<https://www.architect.io/blog/2021-01-11/gitops-developers-guide/>

Schedule 8/15 @ 6:00 PDT

Post 2:

Fact: #Developers live in Git.

Fact: Businesses need manageable and automatable approaches to CI/CD and #DevOps.

So, it should come as no surprise that developers love GitOps. Which is why we've put together this guide to GitOps in practice:

<https://www.architect.io/blog/2021-01-11/gitops-developers-guide/>

Event Driven Architecture

Schedule 7/21 @ 6:00 AM PDT

Post 1:

Event-driven architecture solves problems with fault tolerance, availability, and coupling of #microservices. But what even is an event, and how do you leverage event-driven design in an application?

We answer these questions, plus do a little mythbusting about event-driven design!

Let's dive in: <https://www.architect.io/blog/2021-02-09/event-driven-architecture/>

Scheduled 8/24 @ 6:00 AM PDT

Post 2:

Events make applications fault-tolerant since they don't have to wait for subscribers, but how does that functionally work? What ✨magic✨ is it that allows these events to “complete” even when the subscribers are down or otherwise unavailable?

The answer is surprisingly simple:

<https://www.architect.io/blog/2021-02-09/event-driven-architecture/>

#Programming #DevOps

Scheduled 9/26 @ 6:00 AM PDT

Post 3:

Myth: Event-driven architecture decouples microservices.

Fact: <https://www.architect.io/blog/2021-02-09/event-driven-architecture/>

#APIDevelopment

Scheduled 7/25 @ 6:00 AM PDT

Getting Started Docs

The problem: An endless landscape of operator-focused tools

The solution: A #developer-focused framework that allows you to build and extend cloud services like never before.

Architect is a self-service developer platform for cloud-native #applications and the best way to learn how it works is through practice. We've got you covered with a simple set of steps you can follow to get you started: <https://docs.architect.io/getting-started/introduction/>