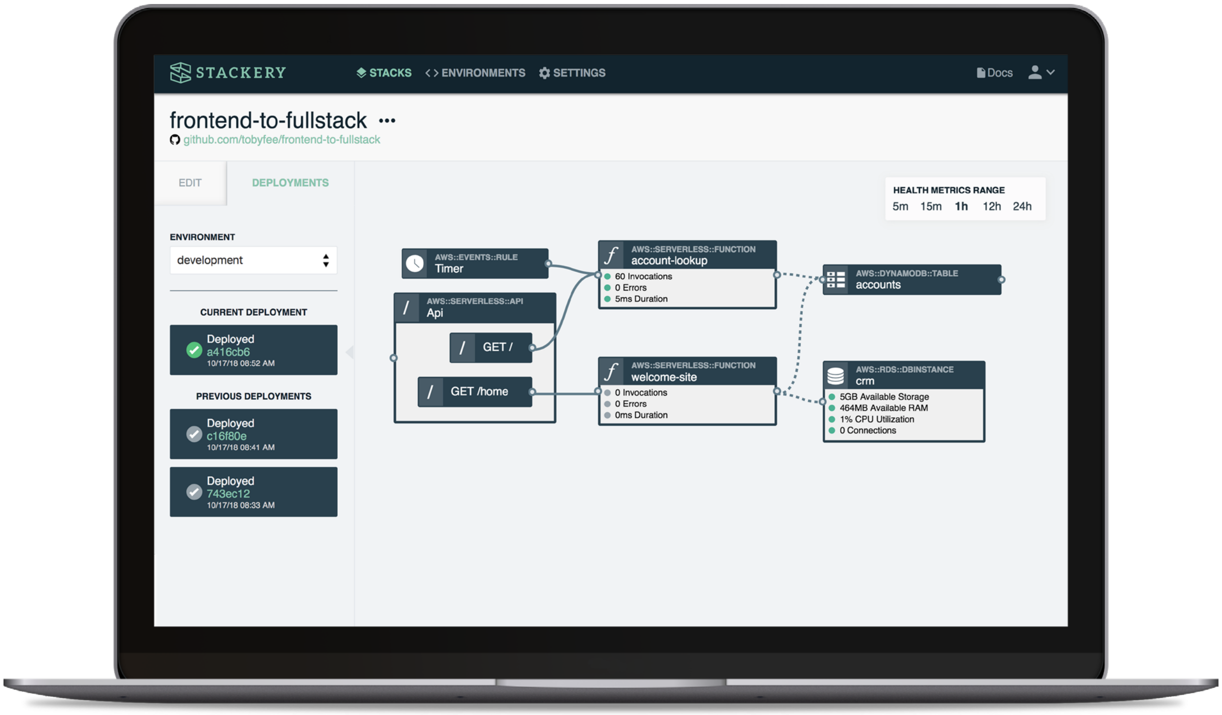
SERVERLESS FOR ENTERPRISE TEAMS

THE THREE CRITICAL STEPS FOR SERVERLESS ADOPTION FOR TEAMS THAT NEED TO SCALE QUICKLY.

STACKERY

There are many amazing advancements happening right now in the world of microservices. At the heart of a lot of these is the adoption of serverless computing. Moving to serverless infrastructure requires changing not only the way you manage your data and code, but also how you plan and execute projects as a team. Here are the three critical steps serverless teams have taken to continue delivering results at scale.



Continual Focus on Business Objectives

At its core, serverless is all about removing operational distractions and infrastructure obstacles that prevent teams from building applications as fast as they would like. Any time engineering organizations adopt a new infrastructure platform, it can be easy to get distracted by operating the platform while losing sight of the need to deliver on business objectives.

When adopting serverless, the primary business objectives are often time to market for a new capability or refactoring a service delivered by an existing monolith to solve performance or scaling issues. Whether that goal needs to be met in the short-term or long-term dictates how the new serverless platform is adopted. If the performance goal is needed in the short-term, then it might be best to start by peeling functionality off the existing application monolith and rebuilding it as new, high-performing web services. If the performance goal is long-term, then it might make sense to start by planning a fundamental architectural shift from the ground up.

The beauty of serverless is the ability to define the resources, the intent of the architecture, and the business logic at the same time, instead of the typical loops found in traditional workflows. The end result of an organizational shift to serverless should always include a substantial shift in the balance of devops tasks. By nature, serverless is meant to allow developers to focus on the “dev” side of devops while reducing the time spent on “ops” as much as possible.

To enable developers to visually define the architectural intent across basic and advanced Amazon Web Services (AWS) while cohesively producing infrastructure configuration code, manage operational namespacing, and enforcing security and instrumentation policies, Stackery built the Operations Console. With Stackery, engineering leaders can easily lay a solid foundation for teams to build serverless applications quickly and focus on primary business objectives.

The New Serverless Workflows + Existing Tools

The adoption of serverless should disrupt engineering teams' workflows as little as possible while saving a substantial amount of time on antiquated ops tasks. With serverless, teams are able to embrace service-oriented architecture on a platform that is built specifically for the task. Gone are the days of imaging a VM and then installing half a dozen dependencies before being able to migrate application code. Gone are the days of container orchestration and splitting up application code across multiple containers.

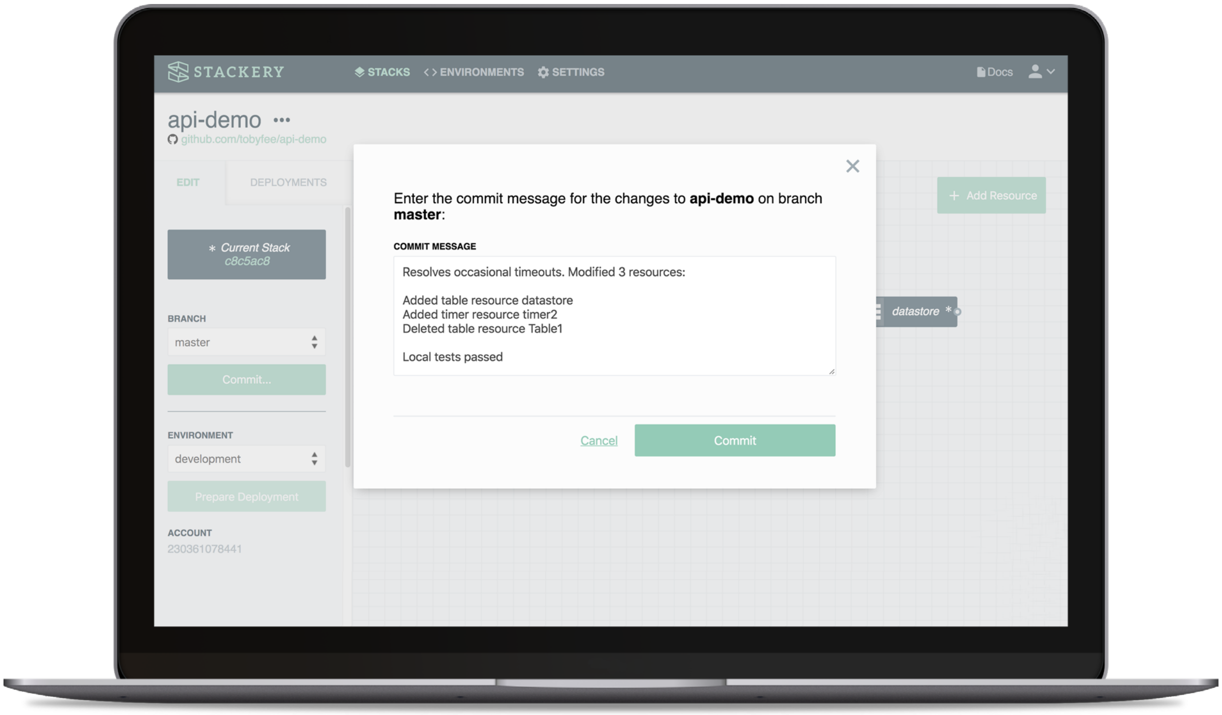
With Stackery, serverless teams can write application code directly to their serverless cloud environments, backed up by the Git-based version control of their choice. In addition, all of the infrastructure code that is used to provision and configure CloudFormation resources is versioned and backed up to GitHub as well. This is a big step forward for engineers that traditionally have had very little visibility into the environment management side of their projects. With Stackery, every environment management change made by anyone on the team can be seen in the Stackery Operations Console and the project repository.

Stackery allows large teams to build, merge, deploy, monitor and scale their serverless architecture all from the same dashboard. Stackery's built-in infrastructure monitoring allows teams to easily understand what resources are taking the largest workloads and find performance problems. This allows teams to adopt serverless environments and greatly increase developer velocity without sacrificing visibility or needing to re-think existing workflows.

Scale Teams, Environments, and AWS Resources

Serverless application teams encounter scaling challenges as the number of developers committing changes, the number of dev and test environments, and the number and consumption of AWS resources all increase.

- **Personnel:** When additional engineers are added to a team, how quickly can they ramp-up, coordinating changes, and collaborate?
- **Environments:** Do architectural changes in test require coordinating disparate Cloudformation templates for prod or can it be done visually and automatically?
- **AWS Resources:** Basic serverless building blocks scale consumption well, but what happens when advanced services, access controls, and databases are added?



* Current Stack
c8c5ac8

BRANCH
master

Commit...

ENVIRONMENT
development

Prepare Deployment

ACCOUNT
230361078441

Enter the commit message for the changes to **api-demo** on branch **master**:

COMMIT MESSAGE

```
Resolves occasional timeouts. Modified 3 resources:

Added table resource datastore
Added timer resource timer2
Deleted table resource Table1

Local tests passed
```

Cancel

Commit

Stackery has been specifically built for teams that need to scale quickly in all three of these dimensions. Fast-moving teams with well-implemented architecture are able to solve more than one of these problems at the same time.

With Stackery, managers can monitor developers' IAM (Identity and Access Management) privileges so that each team member has access to the correct resources in the correct AWS region. Engineering managers also have complete visibility into every infrastructure change through the Stackery Operations Console and the Git logs. This greatly improves the personnel ramp-up time.

The Stackery Operations Console gives every team member complete visibility into what resources are setup in AWS and how they interact. This view offers an architectural diagram that shows, in complete detail, every resource involved in each microservice across dev, test, and production environments. This visibility helps teams to instantly get a complete view into changing architecture as new features are implemented.

Stackery also displays resource-based health metrics provided by real-time data directly from Amazon CloudWatch. With every public release, teams can see exactly which functions are being invoked, with what frequency immediately after code is deployed. This infrastructure monitoring allows engineers to immediately address performance issues and application timeouts before they affect users.

More and more enterprise teams are adopting serverless infrastructure to support their goals of robust and high-performing applications. Stackery enables engineering teams to build applications quickly with complete visibility and control over their serverless infrastructure.

Sign up for a free trial today to learn how you can build amazing serverless applications with Stackery.