

Serverless Applications Impacts on Testing and Development

Sample

Written for: Engineering Staffing Marketing Blog

(Hire Our Team Members)

Serverless applications are everywhere now; and its impact on software architecture, development, and testing is transformative.

What is Serverless

Serverless applications is an interesting name, that really has less to do with the application, and more to do with the technology hosting and storage of the application. Serverless applications do make use of servers, it's just that they use them differently than in the past.

If you consider an application to be a product, activity, or service, you can in turn also think of the server as the house in which that product, activity, or service is homed. In traditional server systems, that house is static, probably like your house, or mine.

In the current "Serverless" system, you can have that same product, activity, and service, but the house can change as the needs grow or shrink- like adding a room when you need more space, or renting that room out when space is not being used.

Serverless technology has benefits for both the server hub, and the producer of the application. Applications using serverless architecture only pay for services when actively using those services- as in executing a process.

Let's Take a More Technical Look

The most well-known and understood advantage and selling point of serverless computing is that it economizes the use of cloud resources. Serverless providers only charge for the time that code is executing, maximizing the function and profitability for both the provider and developer. Interestingly Serverless has also increased stability due to spinning services/instances as needed and having redundancy built into the system.

The process of moving to serverless and the numbers of applications and services that have moved to serverless is a testament to its function, and its existing economic incentives.

Additional interesting serverless strengths are further costs reduction when multiple applications share common components, and in defining workflows.

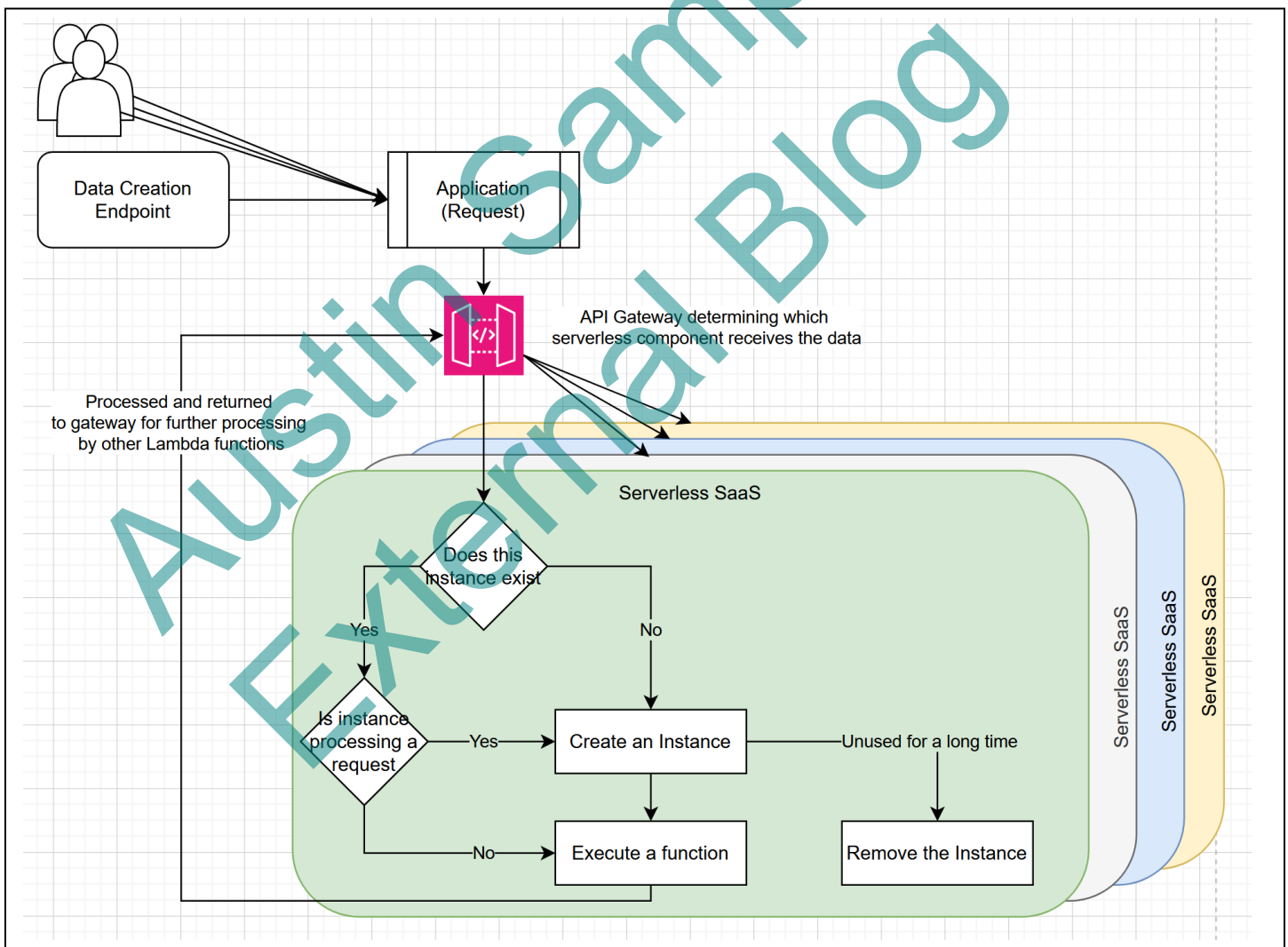
Current thoughts on defining and describing serverless include calling it Event Driven, or Function and a Service (FaaS) protocol. Serverless architecture is best utilized to process events, or discrete chunks of data generated as a time series.

How it Works

Data arrives at the application, (via human or endpoint), and the architecture incorporates an API gateway that accepts the data and determines which serverless component receives the data.

Regardless of which host is being used for the applications serverless architecture, the runtime environment will pass the data to the component, where it is processed, and returned to the gateway for further processing by other runtime functions, or returned to the user completed.

1. Application Development.
 - a. Developers write code, and deploy it to their cloud provider.
2. Cloud Host
 - a. Application code is hosted by the cloud provider, and homed in a fleet of servers.
3. Application use
 - a. Requests are made to execute the Application code.
 - b. The cloud provider creates a new container to run the code in.
 - c. The container is deleted when the execution has been completed.



Considerations

There are challenges and Serverless architecture doesn't work in all things.

It's important to keep in mind that serverless systems are not intended to become complete application. Successful use of serverless requires a separation of data input from computing actions.

Serverless weaknesses can be found in that it doesn't operate as efficiently or successfully if there are longer computation times, or when persistence of data is required. There are some work arounds for this, but they can be problematic. For example, the process takes too long, and serverless will then stop, and require a cold start- it simply may not work for that longer time period. The fix could be to make lots of little computations, that when broken apart, are fast enough to work well in a serverless environment; but the amount of coding time and rebuilding by developers can be prohibitive.

Follow that with a look into testing and quality process, and you have a black box that becomes challenging if not impossible to fix because all of the different pieces are separate. If you fix one, what does it mean to the others. The loop for QA on this could reduce or even remove any cost benefit for being serverless. Programmers and Dev teams will have to add to their workflows, additional steps to recognize, predict, and plan for faults. Training in understanding of the runtime functions, or methodologies for external storage of persistent data may be required.

Lacking persistence of data is another area that will inhibit or transform the both development and testing methodologies. If you do end up with an external method to emulate persistence, is it a security risk? This is a work around that requires further research.

Serverless may require re-designing an application architecture to separate data input and storage from data actions. Serverless components tend to do back-end execution, over front end or user interface operations. Most often there is a separate UI layer or other data input mechanism, that then triggers data into the serverless end; a separate and distinct input from the serverless architecture.

Serverless is Stateless (Lacks Persistence)

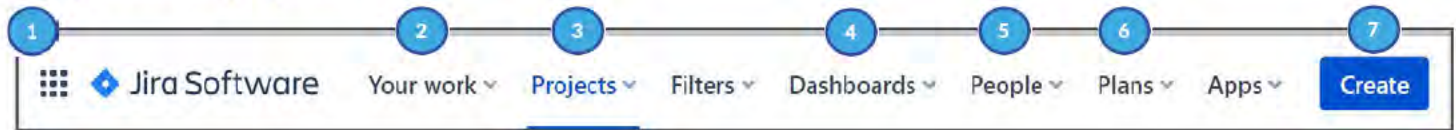
There is one more unique feature of serverless functions – they are stateless. Serverless functions accept input, they process that input in some way, and then return a result. They have no local or persistent storage, and this is by design. In fact, the entire architecture is conceived to have no persistent storage.

Impact of Persistence on Testing and Development

This has several implications for testing. First, it means that you can't test a Lambda function at a time. Testers usually rely on testing a workflow, so that the results of previous steps are available to subsequent steps in the process. On the other hand, it's difficult to test and validate individual functions in isolation. This is thinking about unit versus system testing.

The lack of persistent storage also means that for processing application, developers will need a few steps further.

Navigation Bar | Options

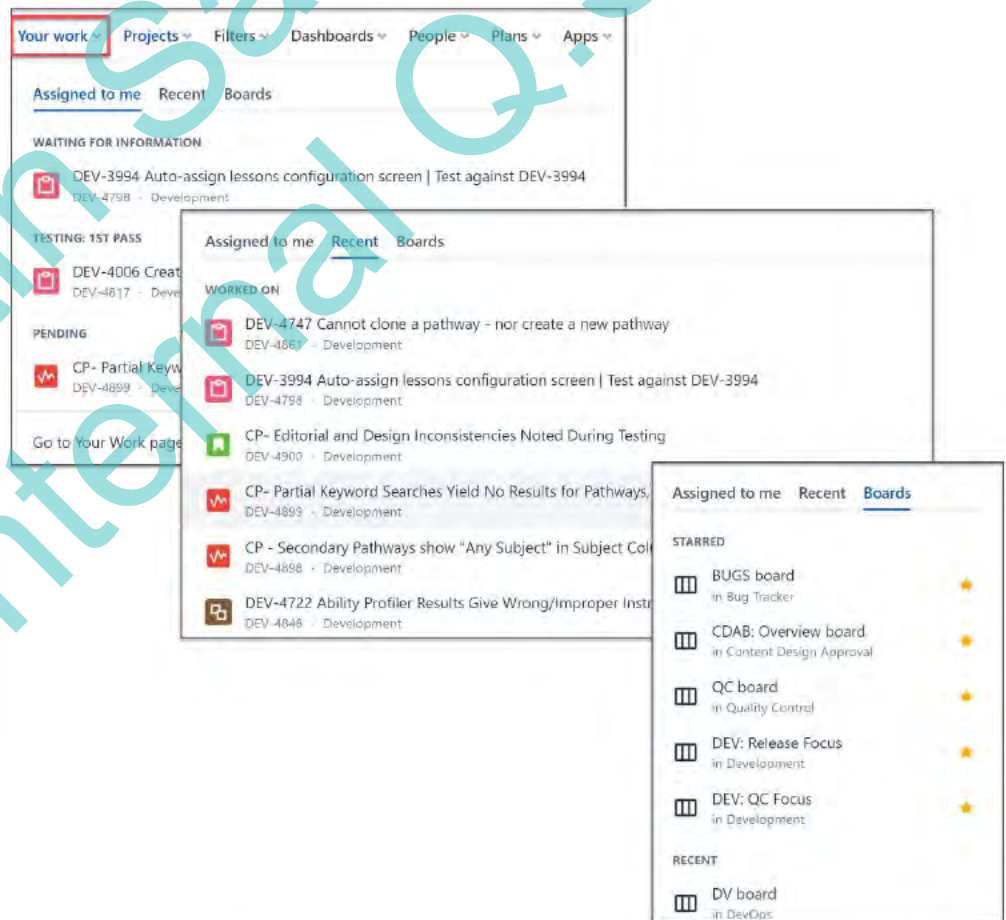


- 1 **Product Switcher** Let's you switch to other Atlassian products like Confluence
- 2 **Your Work** Displays your assigned issues, projects, and recently visited boards
- 3 **Projects** Displays a list of your Projects and the ability to view all Projects
- 4 **Dashboards** Permits you to create, star, or view relevant Dashboards, such as Bug Tracker
- 5 **People** Permits you to view frequent collaborators, other teams, and to start one of your own
- 6 **More** Allows you to view and select enabled Plans and Apps, such as Tempo for timekeeping
- 7 **Create** Allows you to create new issues, such as Bugs and Stories

Additionally, the main **Search** bar on the top right provides Jira site-wide search with predictive text and keyword capabilities; it also displays recent views and allows you to switch to Confluence with a single click.

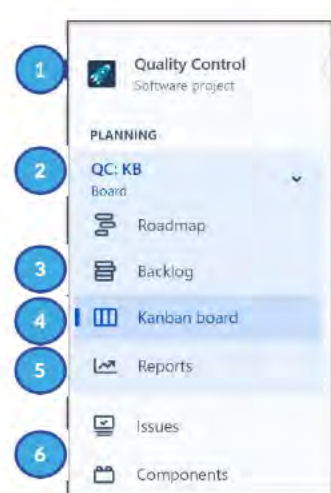
Your Work Menu

One of the menus you will use most frequently is **Your work**, which permits you to view issues assigned to you, issues you have visited recently in Jira, and Boards that you have starred and/or visited recently.



Tip: When you open a Project Board, look to the upper right. You will see a star ☆ favicon that you can select to find the Board easily.

Project Sidebar | Options









- 1 **Project Name** Displays the project you are in—here, it's Bug Tracker, but we have Projects for CSM, SSOR, etc.
- 2 **Board Switcher** Allows you to click a drop-down to switch among various boards in a project, or search boards
- 3 **Backlog** Displays issues that have not yet been started
- 4 **Board View** Displays the selected board
- 5 **Reports** Offers a selection of reports that you can create
- 6 **Lower Menu** Offers options in the lower menu such as **Issues**, which lists all issues within the project and permits you to filter views; other items are for advanced use

Issues | An Overview

Remember that **Issues** are individual work items within Jira. You may also hear them referred to as “tickets.” Issues might be Bugs, Stories, Collateral Requests, Content Ideas, Tasks, and so on.

Issues may be standalone items, or they may be part of a larger effort, such as an Epic, where they may be associated with related issues.

Different issue types even have their own icons to help you gain insights at a glance. Here are some that our teams use.

-  The Task icon is used for SSO/Rostering
-  Epics indicate large projects that comprise several issues
-  The Test icon is used by Quality Control
-  The Story (green) and Bug (red) icons are used by Development
-  The Content Idea icon is used by Customer Success Management
-  The Release Case icon represents the final stages of testing and merge before an issue is resolved and released to the live environment

Try It Yourself

- Navigate to the **Bug Tracker** project board from the link provided on page one.
- Try searches from the main **Search** bar (top right) and basic **Search** from the board itself.
- Open and study a Bug ticket (issue); try clicks of dropdowns to see what you find.
- Select **Projects** and pick another Project board to view. Open and study an issue.
- Select **Your work** and try the different tab views; Click the **Go to Your Work** page and familiarize yourself with its layout.

Sample Internal Learning

Onboarding for Team Member -> Learn the product

Welcome to OurProduct

This scavenger hunt is a fun, hands-on way to explore the platform. It also provides links to additional content and articles that you may find helpful as you learn about OurCompany, OurProduct and our customers.

Note: This scavenger hunt includes several activities to familiarize you with OurProduct. As you complete these activities in our live environment, keep in mind that your usage, assessment results, and completed tasks will be visible to customers and those for whom you do demos.

Before You Begin

To complete the scavenger hunt, you must log on to OurProduct. You will need login details for two user accounts available for this exercise.

- Your personal login information for your OurProduct account, which is an admin user role
- A student account for John Wayne, using the login details below.

Login Details

- Your personal OurProduct login as **yourname@OurProduct.com**, using the password provided
- Student login for username john.wayne and password john.wayne

NOTE: If you do not know your personal login details, check with your team lead or look in your designated password software, such as LastPass.

Begin the Hunt | Junior Role

Hint: We have purposely left some navigation cues out of these instructions so that you can experiment and familiarize yourself with the platform. Happy Hunting.

- 1) Log on to <https://www.ourproduct.com> as user *john.wayne* with password *john.wane* (using the SuperDuper license)
- 2) Select **My SuperTasks** from the main menu bar. The Welcom to My Tasks page displays.
- 3) Begin these assignments:
 - a. Your Interests (Nia)
Complete all components of this taks.
 - b. The Awesome Sauce (Nia)
 - c. Building things (Nia)
- 4) Find **Topics** and complete these tasks:
 - a. Watch 8-10 topic videos. As you visit, be sure to save at least 3 favorites.
 - b. Choose your favorite Topic, and revisit to complete the following:

- i. Add the topic to **My Playlist**
- ii. Complete a **Suprise Survey**
- iii. Find three **Sources** and save them as favorites.
- iv. Find three **Exercises** and save them as favorites.

HINT: You may have to adjust search filters to return results

- 5) Find **My Sources** to view your topic selections. The three you favorited should show up.
- 6) Find **My Exercises** to see your selections. The three you favorited should show up.
- 7) Find **Tools** and complete these tasks:
 - a. Complete 5 reviews
 - b. Complete a **Post Task Plan** and **Save**

Great start! Now, take a look at the Admin view of OurProduct.

Begin the Hunt | Admin Role

- 1) Log on to <https://www.ourproduct.com/login/testing/admins/> using your personal admin login.
- 2) Determine how to navigate to **SuperTasks**.
 - a. Hint
- 3) Find your Tasks. There should be “x” for John Wayne (john.wayne).
 - a. Notice that you can see what John Wayne’s activities completion status.
 - b. Give feedback selection on all three tasks
- 4) Create a new **SuperTask**, and ask the assigned user(s) to visit the SuperSurfer profile.
- 5) Fine **Reports** and complete these tasks.
 - a. Run a report for user Christopher Robin – what were his top three interests?
 - b. Run a report that tells you who spent the most time on OurProduct this year.
 - c. Run a report to determine the name of Christopher Robin’s previous activities.
 - d. Find these details for user Marvin Martian.
 - i. Has Marvin accomplished the surprise survey?
 - e. Find these details for John Wayne

Sample Test Ticket

The following is an example of a test ticket. In addition to the Conditions of Satisfaction, I have included commands within the descriptions that would normally be found on a resources page in a wiki.

Test Ticket Sample

Ticket	Test-XXXX
Dev-NNNN	@theDeveloperHere
Test-XXXX	@theTesterHere
Environment	https://Dev-XXXX.ourproduct.com
Device	
OS	

Conditions of Satisfaction –

Dev-XXXX developed a new carousel that should scroll left to right. Items inside carousel should have links. Actions taken with carousel are trackable using our dashboard on elastic

CoS =

Confirm second carousel on landing page has left/right scrolling behavior

Confirm links

Confirm analytics from elasticsearch – login as dev@ourdomain.com PW: SuperElastic

Data being gathered is time on page, scrolling behavior

- 1) Arrows
- 2) Swipe

Clicking behavior – should include ID

The following test ticket is an example of both initial grooming, and discussion with QC team on what the CoS mean.

Testing: Ticket XXXX

Domain: OurProduct.TestingServerXXX.com

User: TestUser1 PW:TestUser1

Setup :

Open MySQL Server Studio -> open to navigation table

CMD -> SSH into test-XXXX.ourproduct.testingserver

Will be running the **elasticsearch** Set of commands found on **Testing Resources** Page SSH Table

SSH directions found on **Testing Resources** Page

1. ssh ubuntu@dev-XXXX.ourproduct.com
2. cd ourproduct.com/specialsauce.ourproduct.com/
3. Elasticsearch commands
 - a. Php elasticsearch
 - b. php cli.php kpi-update

Browser 1) Open elasticsearch dev environment: test-XXXX.elasticsearch.ourproduct.com

Browser 2) Touch screen enabled device: test server <https://dev-XXXX.ourproduct.com>

Test:

- 1) Navigate to <https://dev-XXXX.ourproduct.com> as external user
 - a. Scroll down to second carousel
 - b. Using arrows move 2 cells to the right
 - c. Using arrows move 1 cell to the left
- 2) CMD: Run SSH command
 - a. ssh php elasticsearch
 - b. ssh php elasticsearch update
- 3) Navigate in another browser to test-XXXX.elasticsearch.ourproduct.com -> Reports
 - a. Check – does the data in elasticsearch include the right/left navigation? Yes = pass
 - b. Does the elasticsearch data include use of arrows? Yes = pass
- 4) Return to carousel as user
 - a. Scroll using swipe of finger – note number of cells here.
 - b. Select one
 - i. Do you navigate to option
- 5) From SSH – confirm data in Elasticsearch
- 6) MySQL Server Studio -> does data include id

SQL queries found on **Testing Resources** Page

Example:

Select top 30 * from kpi_tracking order by created_at desc