PART OF THE SERVERLESS SERIES

EVENTBRIDGE VISUALS

VOL.1

by Serverless Developer Advocate team

Serverless visuals by David Boyne

Serverless Visuals: Small bite sized visuals about Amazon EventBridge

David Boyne

v0.0.0

About Serverless Visuals

Serverless Visuals was designed to help visual learners learn about AWS services. This visual guide is focused on Amazon EventBridge, and goes through various levels of content to help you learn the basics and advanced topics.

This document is organic, every time new visuals are uploaded online this document will be generated and uploaded. (so make sure you keep coming back to check for updated versions).

I truly believe event-driven architectures can transform organisations and I hope you find these notes and visuals useful.

David Boyne

About David Boyne

My name David Boyne and I'm a Developer Advocate at AWS focusing on event-driven architectures and serverless technology. I dive deep into event-driven architectures and create content online to help others. I have also created many open source projects to help you manage event-driven-architectures (e.g. https://eventcatalog.dev).

If you want to learn more or keep up to date with updates feel free to connect with me.

- Twitter: @boyney123
- LinkedIn: https://www.linkedin.com/in/david-boyne/
- GitHub: https://github.com/boyney123
- Website: https://www.boyney.io/
- My Open source projects: https://www.boyney.io/projects

Speaking at Events

If you are interested in having me speak at your upcoming events, I would be happy to discuss the possibility. Whether it's a conference, podcast, seminar, workshop, please feel free to contact me.

Thanks for downloading this content, I hope it can help.

Table of EventBridge Visuals

What are events?

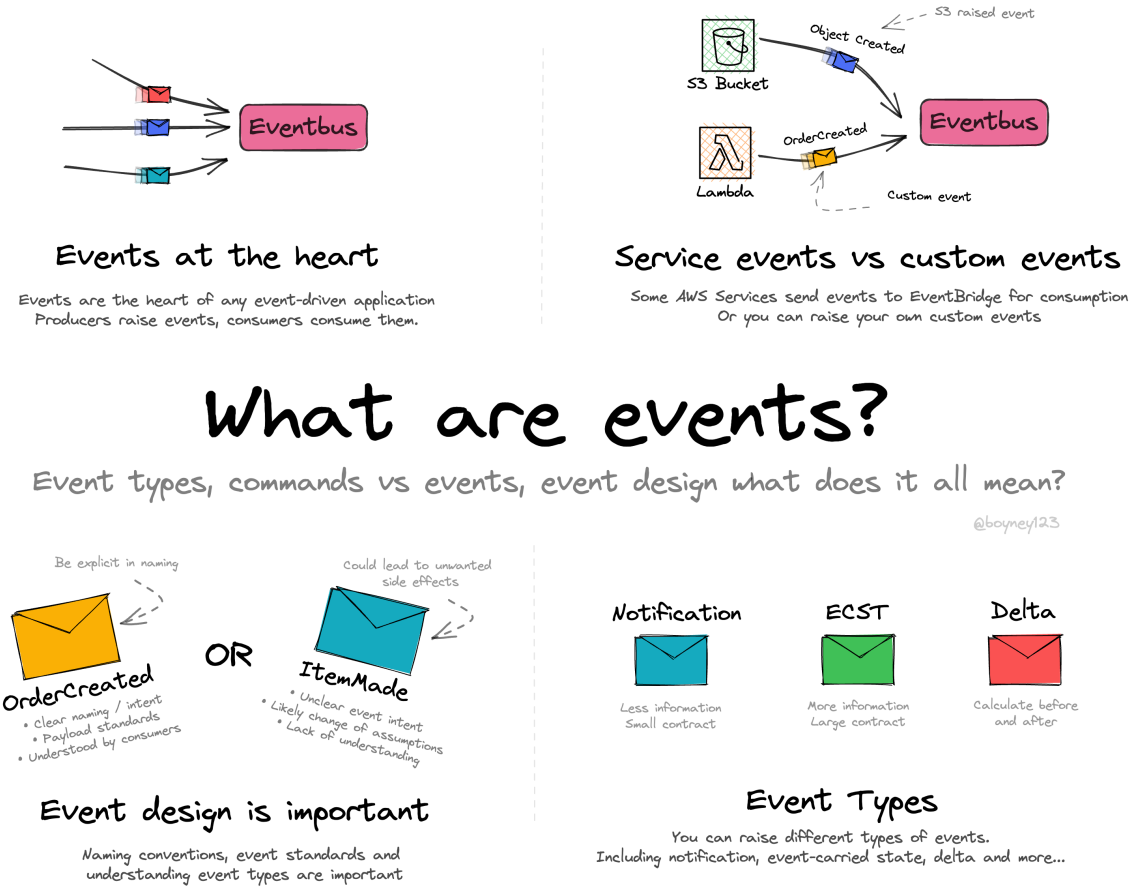

Figure 1: What are events?

Events are at the heart of an event-driven architecture. Think of the event as an envelope of information you can publish across your architecture. What goes into the event payload is up to you, so it's important to consider what information you put in.

Events are often published by producers and downstream consumers react to the events and process them.

Events at the heart

- Events are the core of an event-driven architecture, use events or messages to communicate between services
- Events/messages allow services to remain decoupled, we can use services like Amazon EventBridge to process events for us. We publish events to an event bus and EventBridge handles

configured rules and consumers.

Service events vs custom events

- When you build event-driven architectures on AWS with EventBridge you can consume service events or raise your own custom events.
- Service events allow to react to AWS service events. Example would be when an item is create in Amazon S3, you can setup downstream consumers to react to this event/notification.
- EventBridge also allows you to create custom events. These can be business events or domain events. Raising important business critical events can help you build decoupled event-driven architectures. An example of this would be raising an `OrderCreated` event when a new order has been created, and letting downstream or external systems consume this event and then process the order.
- Remember if you are writing custom events, it's important to consider your event design and best practices.

Event design is important

- When designing events, the name and payload of the events is up to you, so it's important to consider what information needs to go into your event.
- Make sure your naming conventions are implicit. Make sure your event names are clear and the intent of the event can be determined by reading the event name itself. This can help with consumer discoverability.
- Use workshops like Event Storming to help identify events and start to agree on standards within your teams / domains and organizations.

Event Types

- When raising custom events you can choose which type of event you want to raise. These are explicit but you can determine which type of event you want to raise by the information you put into your event
- Notification events are small events, designed to notify downstream consumers. These tend to be small in contract and downstream consumers may request more information from the producer or another API to get information required.
- Event-carried state transfer events tend to have more information inside them. They are rich in content and downstream consumers often have the information they need. Remember when using EventBridge, you can use filters to filter events before they reach downstream consumers, information inside the events can help with this, but understand the trade-offs with your event design.
- Delta events provide consumers information of the state before and after something has happened. An example of this would be a ShoppingCartUpdated event, you can share with consumers the total cost of the shopping cart before/after items are removed. This allows downstream consumers to remain simple and not need to calculate any deltas themselves.

Extra Resources

- EDA Visuals - Explore over 40 visuals about event-driven architectures, learn the fundamentals and advanced concepts. Dive deeper and explore resources linked to every resource.
- Commands vs Events - When you dive into event-driven architecture you will see the terms commands and events. Understanding the differences here is important.
- Event first thinking - It's easy to just raise events and put minimal design thought into your events, you may run into issues later on though. Treat events as first class citizens, this visual can help you understand more.
- Explicit vs Implicit events - Don't let consumers guess what your events do, make clear implicit event naming conventions to help.
- Exploring event types - Dive deeper into event types, understand the tradeoffs between them.
- Awesome EventBridge - A list of resources from blogs, videos and examples all about EventBridge. Feel free to explore or even contribute to the list.

Understanding EventBridge targets

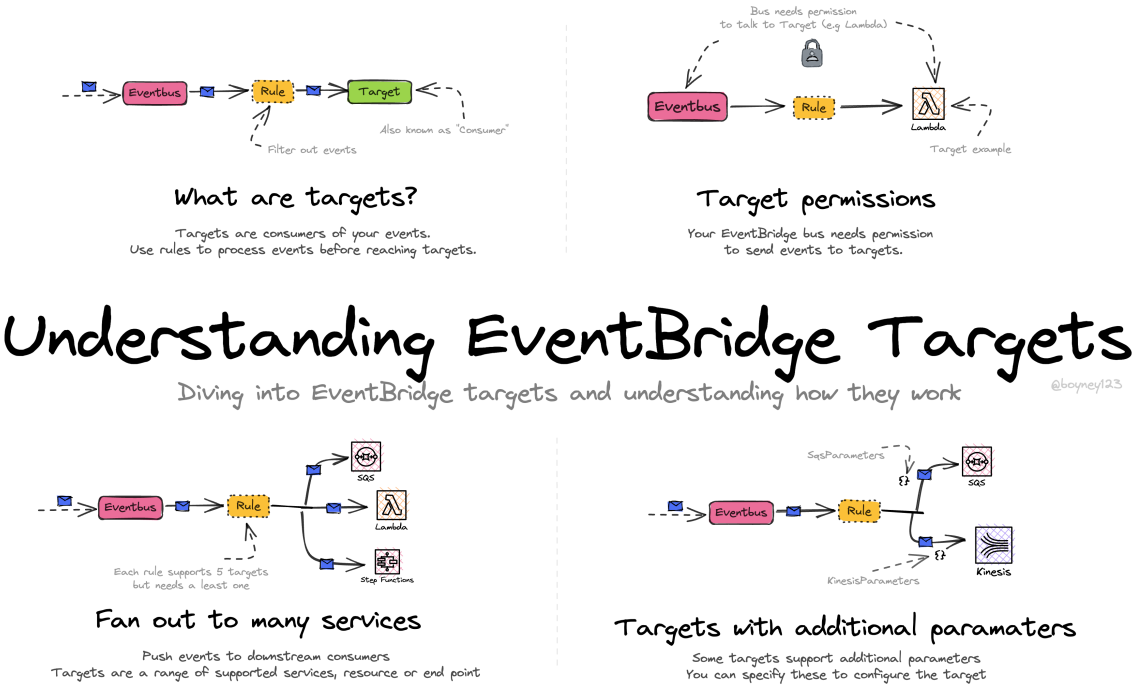

Figure 2: Understanding EventBridge targets

When building event-driven applications, you may want to publish events to downstream consumers. Amazon EventBridge allows you to configure rules and targets to create consumers of events.

Amazon EventBridge allows you to listen to events raised in various AWS Services, and also create and publish your own custom events. Once an event is published to your event bus, EventBridge will process these events and trigger your downstream consumers (targets).

What are Targets?

- When you publish an event, you want a downstream consumer to react. Targets are consumers of your events.
- A target cannot live without an EventBridge rule. Rules process events before they reach downstream targets.
- When an event triggers a rule all of the targets associated with the rule are invoked (unless you have filtering setup on your rule)
- Targets are services, resources or end points you want your event to trigger (read the full list here).

- Targets can also be services in other AWS accounts or Regions. This allows you to setup multi bus topologies within your architecture. Read more.

Target Permissions

- Your EventBridge bus will need permissions to trigger downstream targets.
- To make API calls against the resources you own, EventBridge needs the appropriate permissions. For Lambda, Amazon SNS, Amazon SQS, and Amazon CloudWatch Logs resources, EventBridge uses resource-based policies. For Kinesis streams, EventBridge uses identity-based policies. You can read more here.

Fan out to many services

- EventBridge is a fully managed serverless service that allows you to fan out events to downstream consumers. (also known as publish/subscribe pattern)
- Using events you can notify downstream consumers that something has happened. These producers of events can be AWS service events or even custom events.
- **Custom events are powerful**. They allow you to notify downstream consumers that something has happened. These can be important business domain events, or maybe technical events. When developing your events it's important to understand event design and the different types of events you may raise.

Targets with additional properties

- EventBridge supports 28+ different targets, these can be AWS Services and also external APIS using API Destinations.
- Depending on what target you configure, you may have the option to add additional configuration to your targets.

Extra Resources

- EDA Visuals - Explore over 40 visuals about event-driven architectures, learn the fundamentals and advanced concepts. Dive deeper and explore resources linked to every resource.
- Publishing events without targets - When publishing events, you may not know of downstream consumers, or even have any, but one great practice of EDA is you may want to publish events without any consumers (yet). This visual helps you understand more.
- Using EventBridge with legacy applications - Event driven applications are not just for green field projects, but using events can help you migrate off legacy systems. Maybe Amazon EventBridge can help you here with your migration into the cloud?
- Flow Architectures - Great book by James Urquhart about how events can and will be used to communicate between organisations, when using EventBridge you may want to explore API Destinations being able to trigger internal/external APIS with your events, creating and moving to a flow architecture.

- What are producer and consumer responsibilities? - When building your EDA application your producers and consumers will have their own set of responsibilities, this visual can help you understand more.
- Understanding event types - When you publish events, there are many types of events to consider, this visual helps you understand them and dive deeper.
- Awesome EventBridge - A list of resources from blogs, videos and examples all about EventBridge. Feel free to explore or even contribute to the list.

What are EventBridge rules?

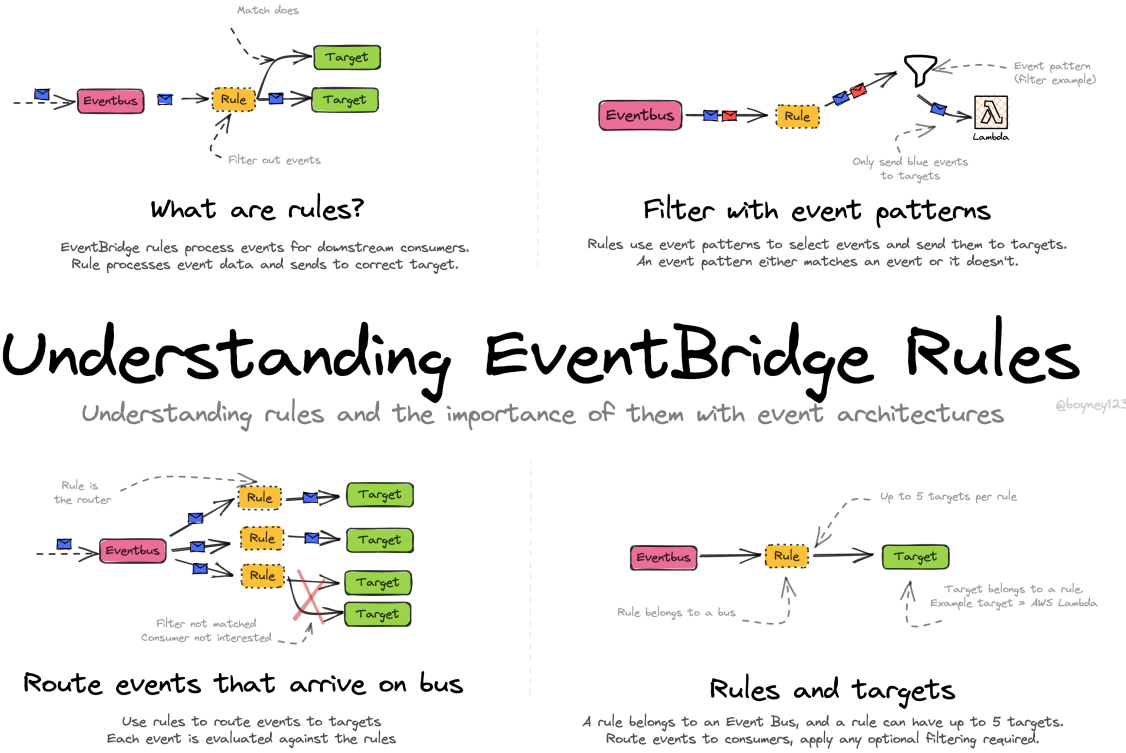

Figure 3: What are EventBridge rules?

Amazon EventBridge rules allow you to route events on your EventBridge bus to targets. Producers can publish events to EventBridge, but they will only be consumed by consumers when rules and targets are configured.

Think of EventBridge rules as the router between your events and downstream consumers.

Your rules understand which consumers to trigger (these are EventBridge targets). Rules may also have filters on them, this allows events to be filtered and only trigger downstream consumers when these patterns match.

What are EventBridge rules?

- Rules allow you to route events that arrive on your event bus to downstream consumers.
- Rules can be configured with filters (using event patterns). Use this to trigger consumers when a particular event or details in the event match your use case.
- Rules can have up to 5 targets (consumers). When an event matches a rule, the downstream targets are triggered in parallel (pub/sub pattern).

- When rules fail to deliver events to a target, the default retry period is 24 hours, until the event is dropped or configured on a dead-letter queue (DLQ).

Filter with event patterns

- Using rules you can filter events to your downstream consumers. Many publishers may be producing events, you can listen to these, filter them (e.g. by name or property in the event), and then trigger consumers
- Filtering is a great way to consume only the events your consumer wants to know about, and gives the ability to be flexible with filtering options.

Route events that arrive on the bus

- Think of rules as routers, they understand what the events are, if they should be filtered and forwarded onto the consumers. Rules control the information if consumers are triggered or not.
- Rules can be enabled or disabled. Keep this in mind if you ever want to turn a rule on or off.
- There are three rule types, `Managed`, `Standard` and `Scheduled`. `Managed` rules are owned by the EventBridge service. `Standard` rules are rules managed by you with pattern matching and `Scheduled` rules allow you to trigger targets on time schedules. If you are interested in scheduled rules it's worth checking out Amazon EventBridge Scheduler.

Rules and Targets

- Rules are configured on the Event Bus, and targets (consumers) are configured on the rules.
- Targets are attached to rules. The rule routes events to downstream consumers (targets).
- You can have 5 targets per rule, each target will be triggered in parallel. This is known at the publish/subscribe pattern. Notify downstream consumers that something interesting has happened.

Extra Resources

- EDA Visuals - Explore over 40 visuals about event-driven architectures, learn the fundamentals and advanced concepts. Dive deeper and explore resources linked to every resource.
- Awesome EventBridge - A list of resources from blogs, videos and examples all about EventBridge. Feel free to explore or even contribute to the list.
- Publish events without consumers? - Remember you can still publish events without any rules or consumers, this might be a pattern you may want to explore. This visual can help you understand more.
- What happens with delivery of events fail? - When building event-driven architectures events can fail to reach downstream consumers, how can you handle that? Dive in and find out more.
- Internal vs External Events - When you have your services/domains you may be raising internal or external events, having rules that are internal to your domain or external by other

teams. Knowing the difference between these can help. Remember to think about what goes into your event payloads.

- Document your event-driven architecture - When you start building consumers, targets, rules and event buses, you may want to consider documentation. There will come a point where your producers and consumers scale, and it can become hard to discover events, schemas, producers and consumers.

Event Patterns: Filtering events with Rules

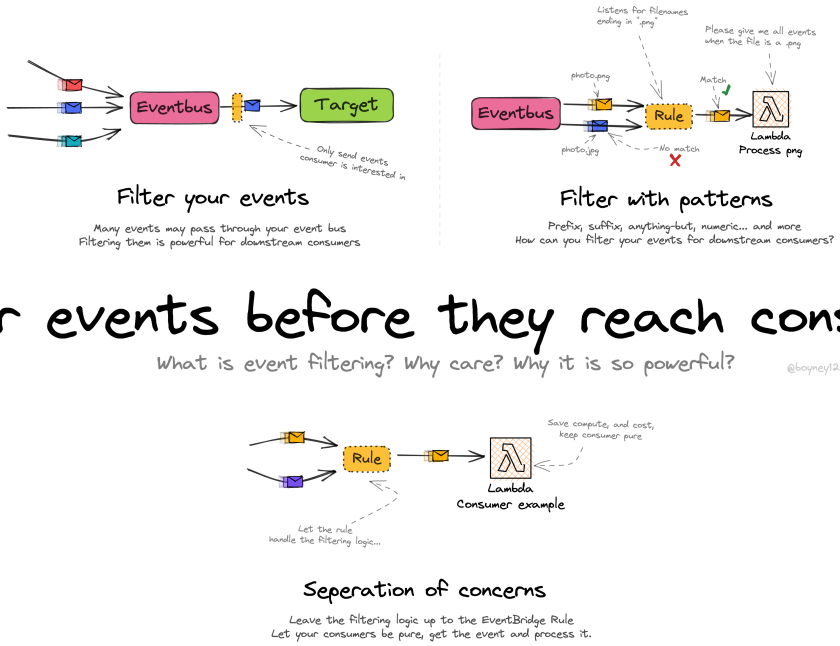

Figure 4: Event Patterns: Filtering events with Rules

When you are building serverless event-driven architectures with Amazon EventBridge, you may want to filter your events before they reach downstream consumers.

EventBridge rules process events before they reach downstream consumers (targets) and use event pattern matching to work out if your consumer should be triggered or not. This is a **powerful feature**, as it allows you to trigger customers only when certain conditions are met.

Filter your events

- Use EventBridge event pattern matching in rules to filter your events, the rules will process the events and only forward events to consumers if the criteria by your filter is matched.
- EventBridge supports a wide range of filters (e.g Preffix, Suffix, anything-but, numeric and much more...). It's worth reading them and understanding what filter patterns you can use to help build your EDA applications.
- One thing to consider is someone has to manage your EventBridge Rules (and filters), this can be your consumers, or for some users this may be your DevOps team. Make sure you have clear responsibility of who manages these rules and filters. The producer should not really care about who is consuming their events (at the technical level anyway, but remember you may care as an organization who is producing and consuming what).

Filter with patterns

- EventBridge supports many filtering options for you. You specify these using "Event Patterns". When events match your specified pattern, the event will be forwarded to your consumer.
- It's quite common to match on source and detail type, these would be the names of your events. For example you can match on `OrderCreated` event (OrderCreated would be the `detail-type` of the event). If your consumers are interested in particular events, for example orders created over $100, you could you filters like numeric matching filter to help.

Separation of concerns

- Let EventBridge rules do the filtering for you vs filtering in your consumers (where you can). Having the filtering logic outside of your consumers allows you to keep your consumers simple and not worry about any filtering logic.
- Any code we write we have to maintain, if the first thing your consumer is doing is filtering EventBridge events, then maybe you can move this logic into your rules, have EventBridge manage this for you and let your consumer focus on the logic it needs too.
- Make sure you have rule ownership, who owns the rules for your event bus? Is it the consumer or someone else (some folks have DepOps teams control EventBus rules)? Have Rule ownership in your EDA architecture, as time goes on this can help you scale your architecture and not lead to unwanted side effects of ownership.

Extra Resources

- EDA Visuals - Explore over 40 visuals about event-driven architectures, learn the fundamentals and advanced concepts. Dive deeper and explore resources linked to every resource.
- Awesome EventBridge - A list of resources from blogs, videos and examples all about EventBridge. Feel free to explore or even contribute to the list.
- What is suffix matching - Understand suffix matching, when you can match values at the end vs the start (prefix).
- What is prefix matching - Understand prefix matching, when you can match values at the start vs the end (suffix).
- Understanding EventBridge Rules - If you want to learn more about EventBridge rules, here is a visual that can help you.
- What are EventBridge Targets? - Targets are consumers of your events, this visual dives abit deeper with resources to learn more.
- Publishing events without targets - When publishing events, you may not know of downstream consumers, or even have any, but one great practice of EDA is you may want to publish events without any consumers (yet). This visual helps you understand more.
- What are producer and consumer responsibilities? - When building your EDA application your producers and consumers will have their own set of responsibilities, this visual can help you understand more.

- Understanding event types - When you publish events, there are many types of events to consider, this visual helps you understand them and dive deeper.

Event Patterns: Prefix filtering

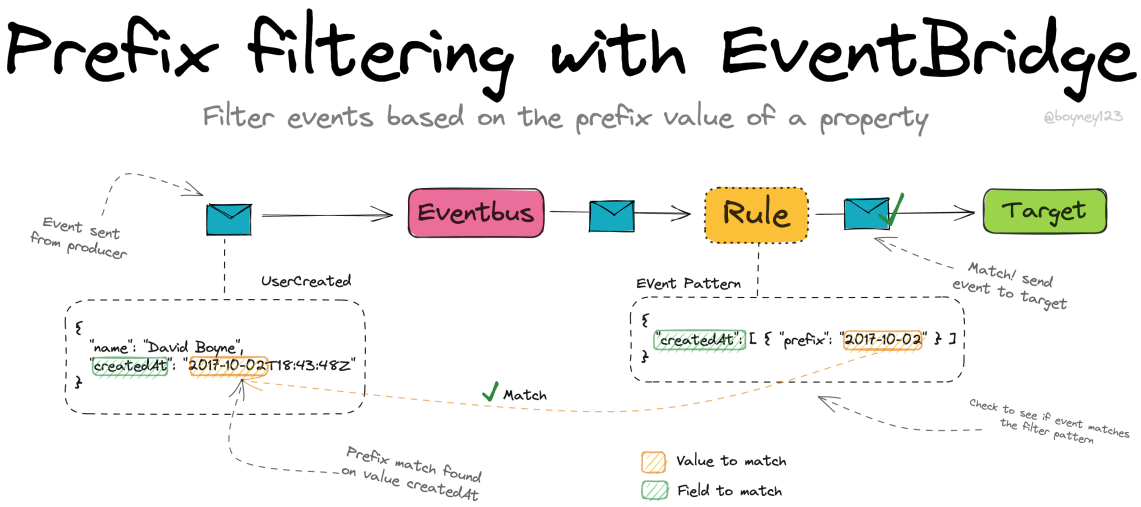

Figure 5: Event Patterns: Prefix filtering

- EventBridge pattern matching is a feature in Amazon EventBridge that allows you to filter incoming events based on specific criteria.
- Prefix matching is a type of pattern matching where you only match events whose specified attribute values start with a given string.

Extra Resources

- EDA Visuals - Explore over 40 visuals about event-driven architectures, learn the fundamentals and advanced concepts. Dive deeper and explore resources linked to every resource.
- Awesome EventBridge - A list of resources from blogs, videos and examples all about EventBridge. Feel free to explore or even contribute to the list.
- What is suffix matching - Understand suffix matching, when you can match values at the end vs the start (prefix).
- Awesome EventBridge - A list of resources from blogs, videos and examples all about EventBridge. Feel free to explore or even contribute to the list.

Event Patterns: Suffix filtering

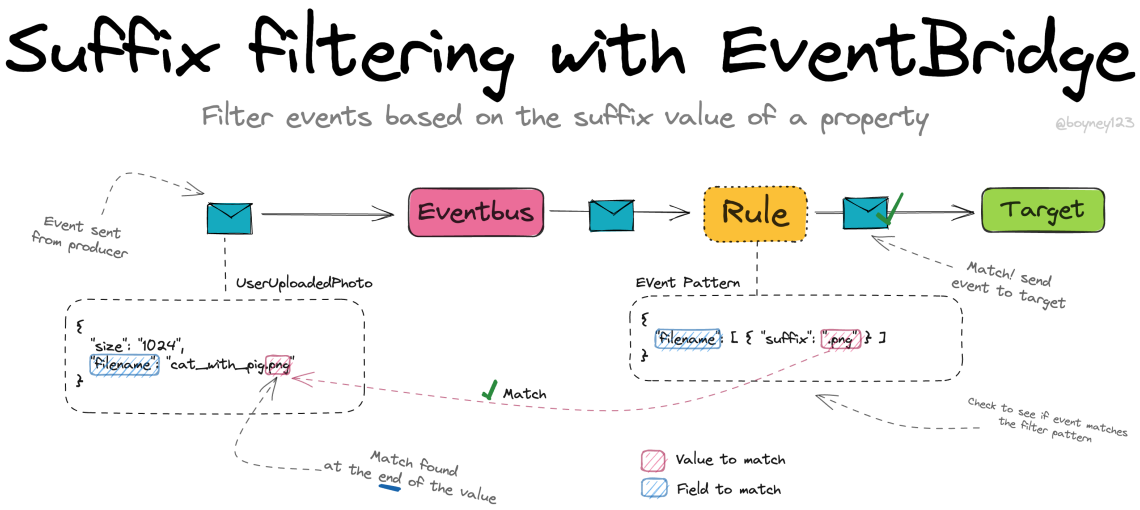

Figure 6: Event Patterns: Suffix filtering

- EventBridge pattern matching is a feature in Amazon EventBridge that allows you to filter incoming events based on specific criteria.
- Suffix matching is a type of pattern matching where you only match events whose specified attribute values end with a given string.

Extra Resources

- EDA Visuals - Explore over 40 visuals about event-driven architectures, learn the fundamentals and advanced concepts. Dive deeper and explore resources linked to every resource.
- Awesome EventBridge - A list of resources from blogs, videos and examples all about EventBridge. Feel free to explore or even contribute to the list.
- What are filters? Why are the important - This visual can help you understand what are EventBridge filters with Rules and why they are important when building EDA applications.
- What is prefix matching - Understand prefix matching, when you can match values at the start vs the end (suffix).

Learn event-driven architecture today

If you want to learn more and get started building event-driven architectures on AWS, here are a few resources that can help you.

EDA Guide on Serverlessland.com Serverlessland.com hosts over 400 AWS Serverless patterns, 60 Step Functions workflow patterns and over 40 serverless code snippets, we also have a new EDA section to help the community learn more about building event-driven architectures. If you want to know more you can visit https://serverlessland.com/event-driven-architecture/intro

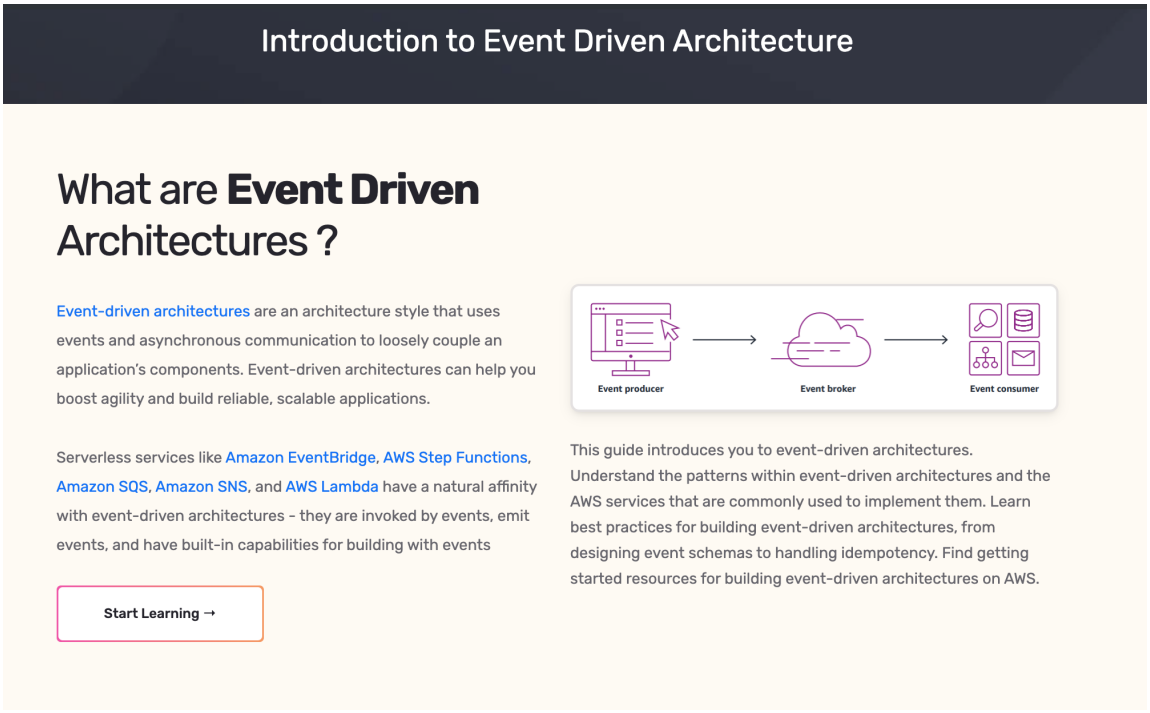

Figure 7: Learn event-driven architecture on Serverlessland.com

30 Days of Serverless and EDA For those who want to get hands on building serverless event-driven-architectures on AWS, we have created a new program for 30 days of Serverless. These are a collection of videos we have put together to help you learn serverless and event-driven-architectures from the ground up. If you want to learn more you can visit https://serverlessland.com/reinvent2022/30dayssls

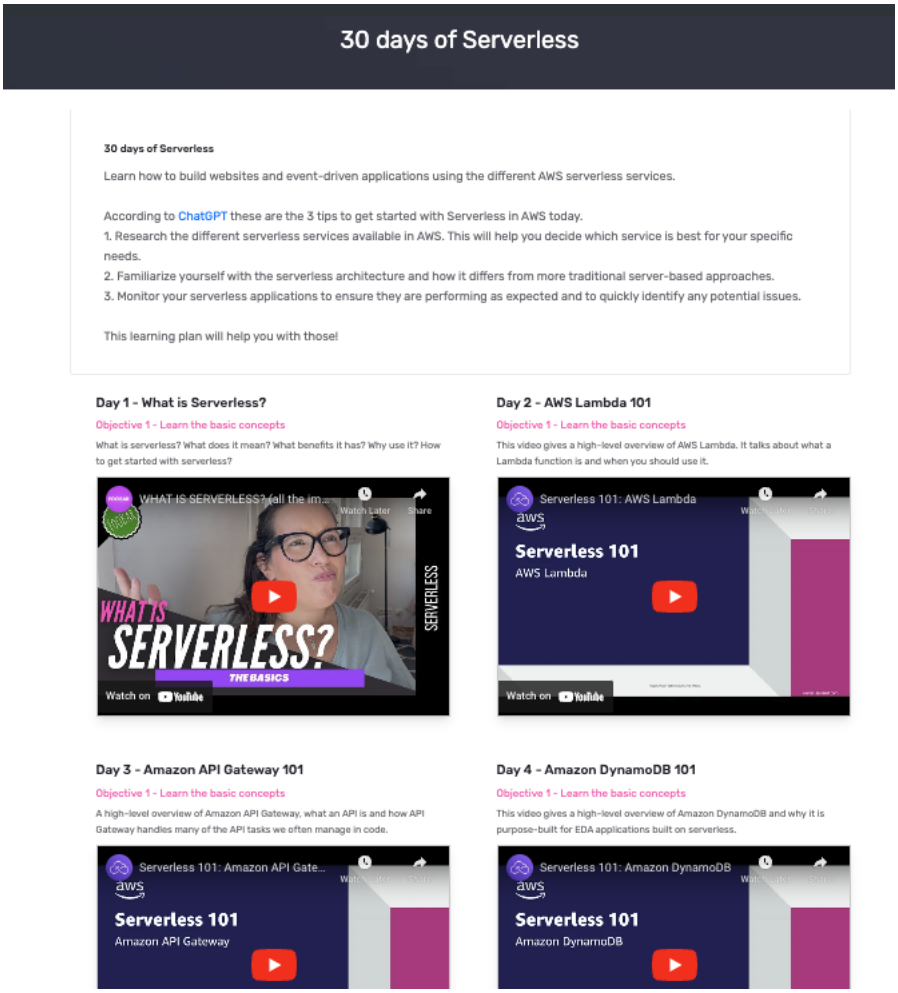

Figure 8: Learn event-driven architecture on Serverlessland.com

Summary

Serverless Visuals are a colletion of visuals to help you learn AWS services. This visual guide is focused on Amazon EventBridge.

You can use the visuals to help you get a high level overview of Amazon EventBridge, and use the resources to dive deeper.

The visuals in this document are from https://serverlessland.com/serverless/visuals/eventbridge and this document will be generated again with every new visual added.

Hopefully you find this content useful, and feel free to connect with me if you want to learn more.

- Twitter: https://twitter.com/boyney123
- LinkedIn: https://www.linkedin.com/in/david-boyne/
- GitHub: https://github.com/boyney123