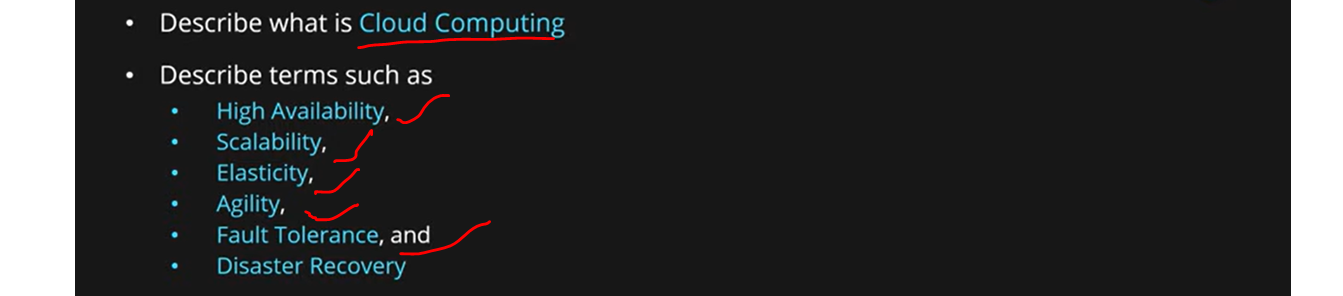
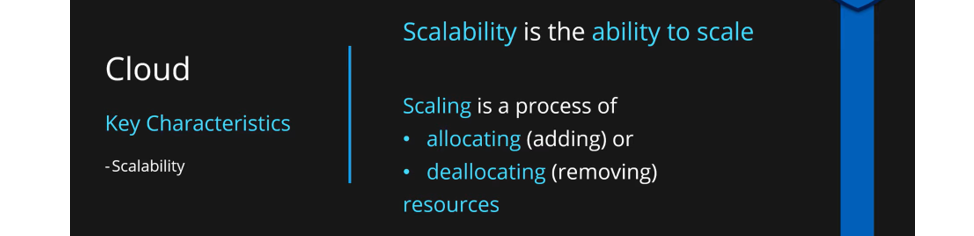
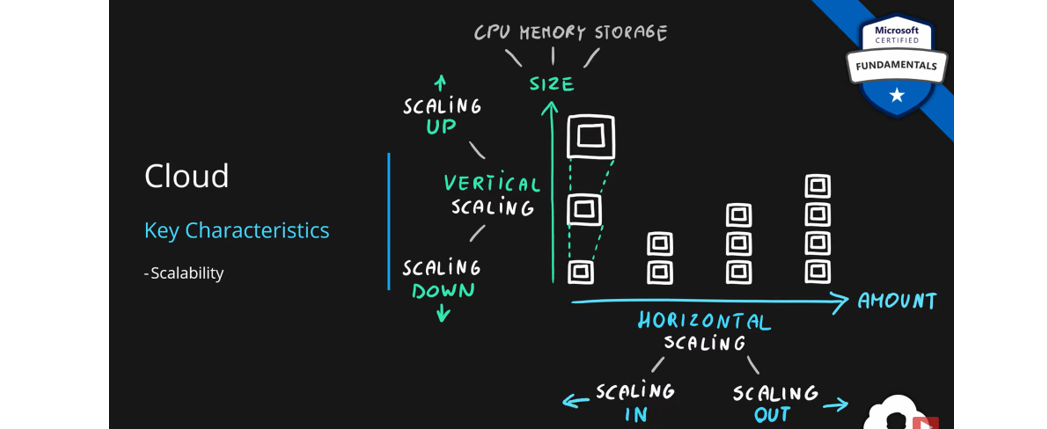
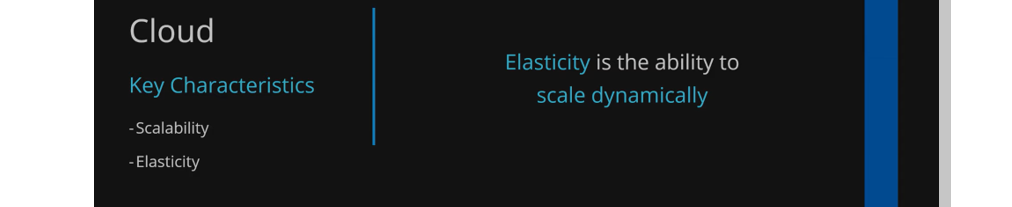
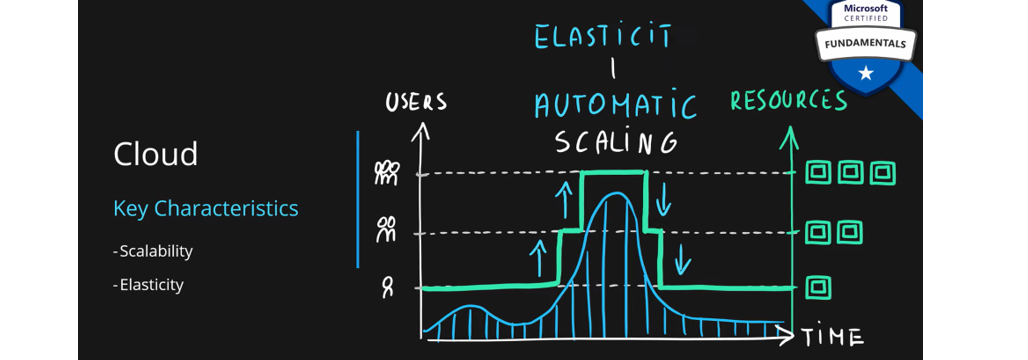
**Cloud Computing and Vocabulary**

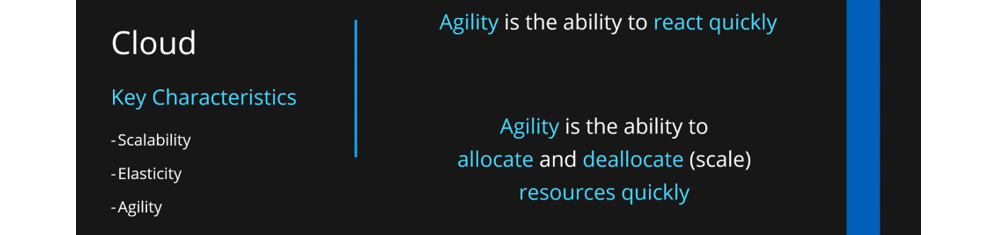


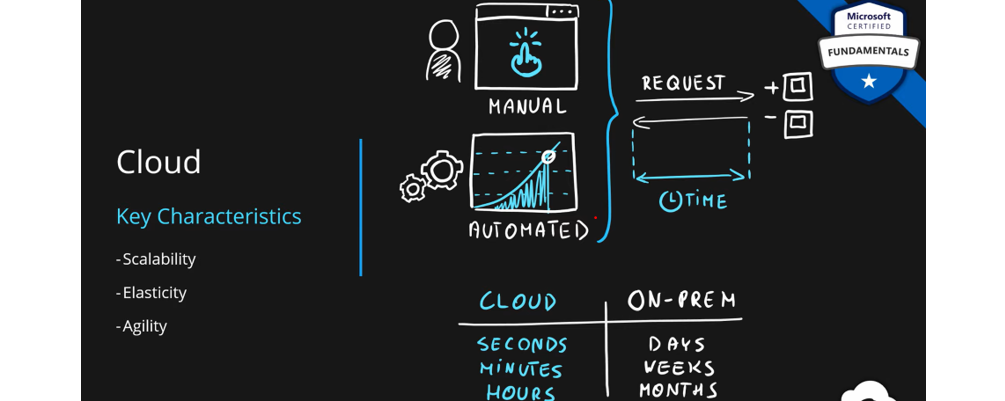


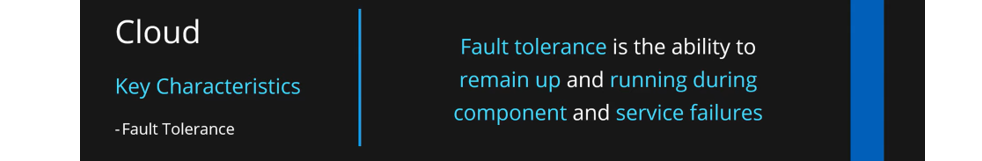


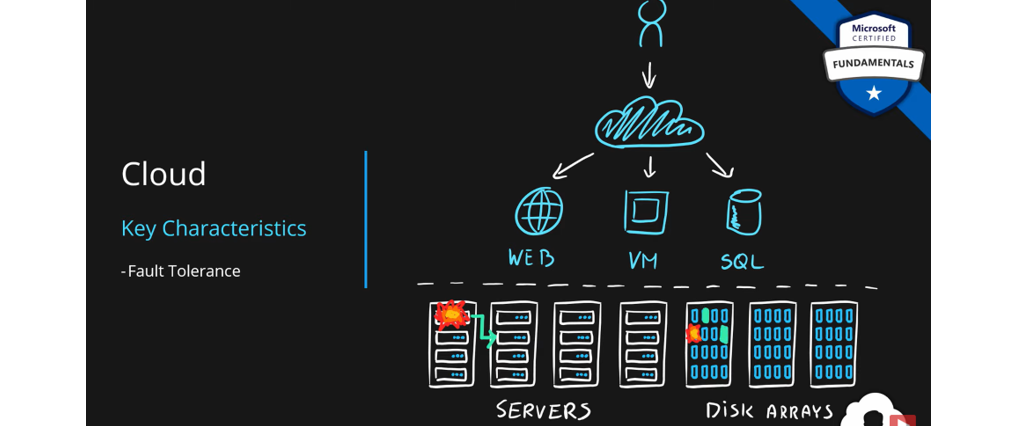


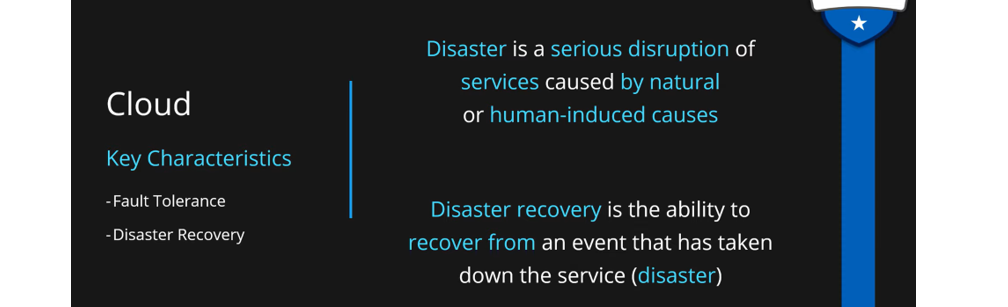


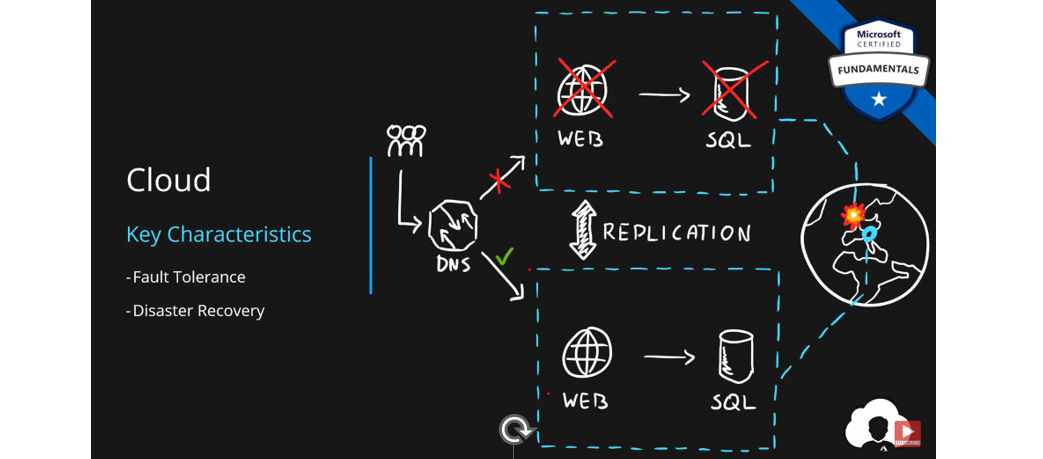


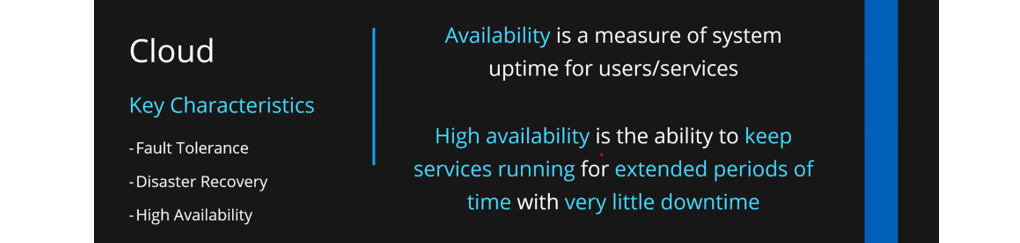


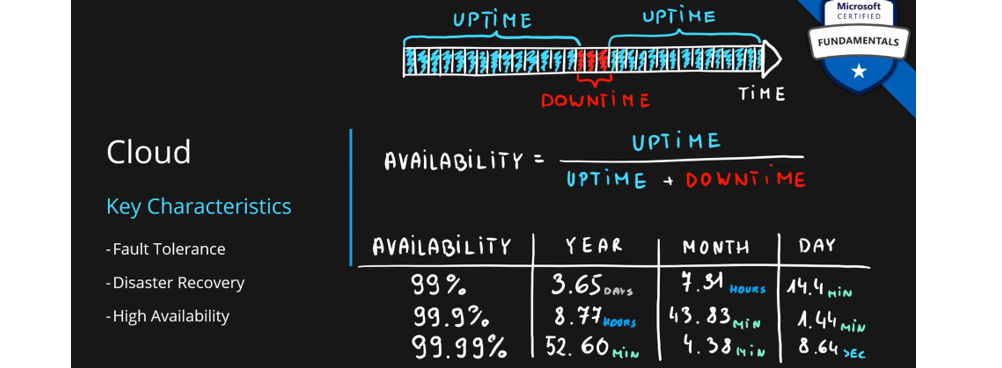


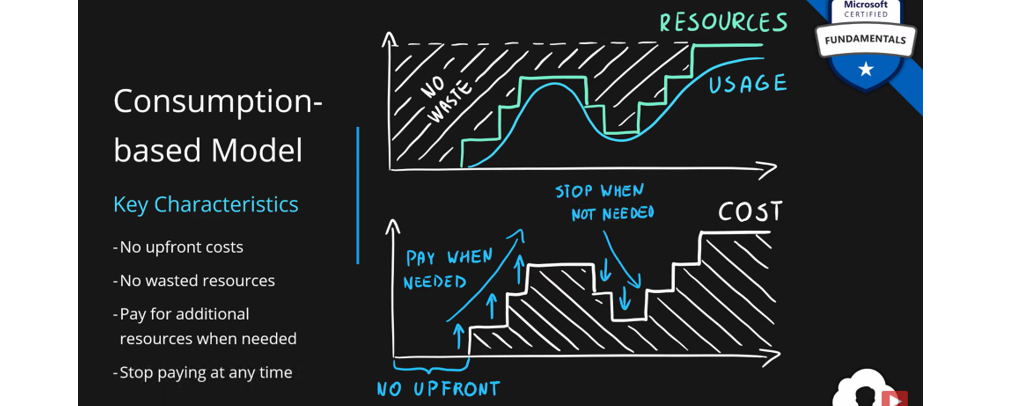






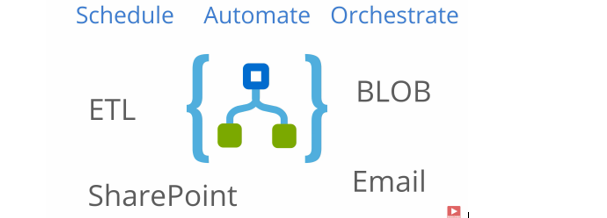






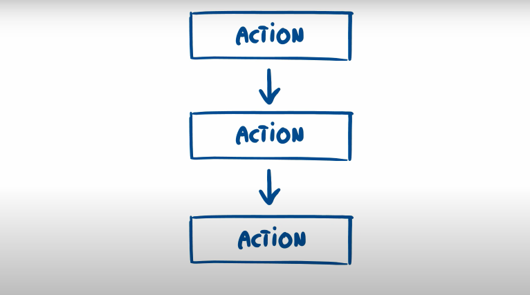
**Azure Logic App**

when it comes to integration across multiple services in Azure, there's actually no better service than Azure Logic Apps.

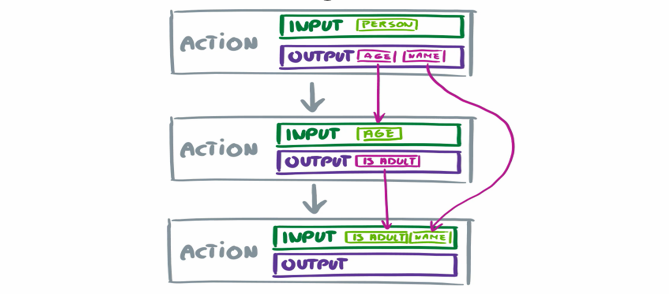


Azure Logic App is a cloud service that helps you to schedule, automate and orchestrate tasks, business processes, and workflows when you need to integrate apps, data, systems and services across enterprises or organizations. What this really means is that you are creating a flow that represent your business flows.

First of all, you always have an action. Every step of Logic Apps is an action. So, it's action followed by an action followed by an action.



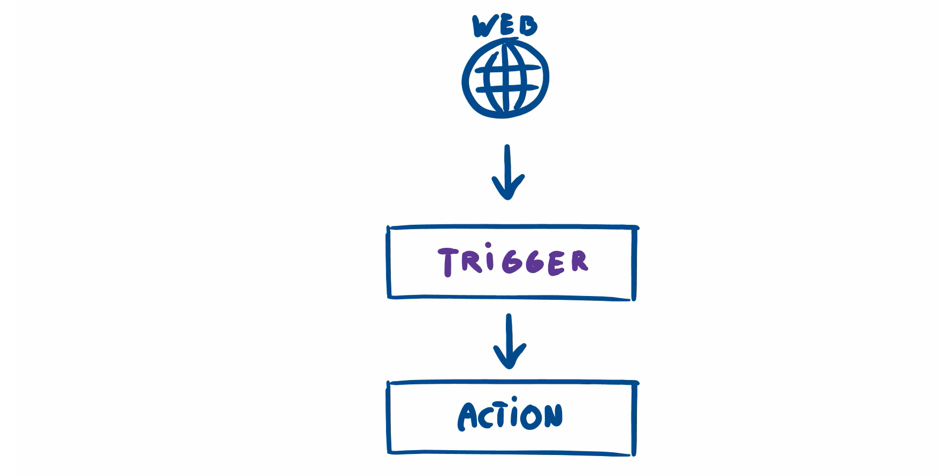
So every action has an input and an output and this goes for every single action. let's imagine for input we got a person object and from the person object, we extracted two properties, age and a name. That's going to be our output in a first action. What's great about this is that we can use now the age as an input to the next step.So now we are able to make some additional logic based on an age. Maybe calculate is this person an adult and now we have additional output, which we can use in a further steps as an input. But this is not where it ends because logic apps allow you to also grab output from previous steps. Like for instance, the name. This is very powerful feature because you are now able to combine multiple outputs from multiple steps and create more complex logic.

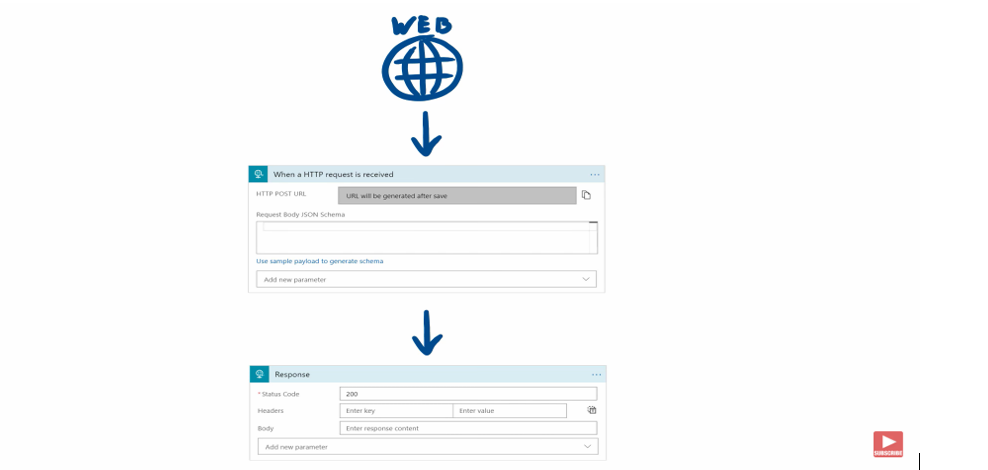


So, we have a certain set of actions, but we need to somehow start those actions. So, in order to start action and let's talk about starting of a logic app. So, first action in a logic app is actually called a trigger because it's a specific action that invokes entire flow. This trigger can be called by multiple sources. Maybe it's a web request. Maybe it's a schedule based trigger. Or maybe it's just an Office 365 event. Or maybe it's just a blob storage event where there's a new file. So, this is actually there's over 200 connectors that you can use for your integrations. There's a lot of things that you can actually use here.



Let's consider simple example with a web request. We have a web request. We have a trigger that is basing on web request and following an action. If you would just move this a bit aside and replace this trigger with a block that looks like this when HTTP request is received and replace the following action by request response, that simply said is the most simple logic app that you can build.

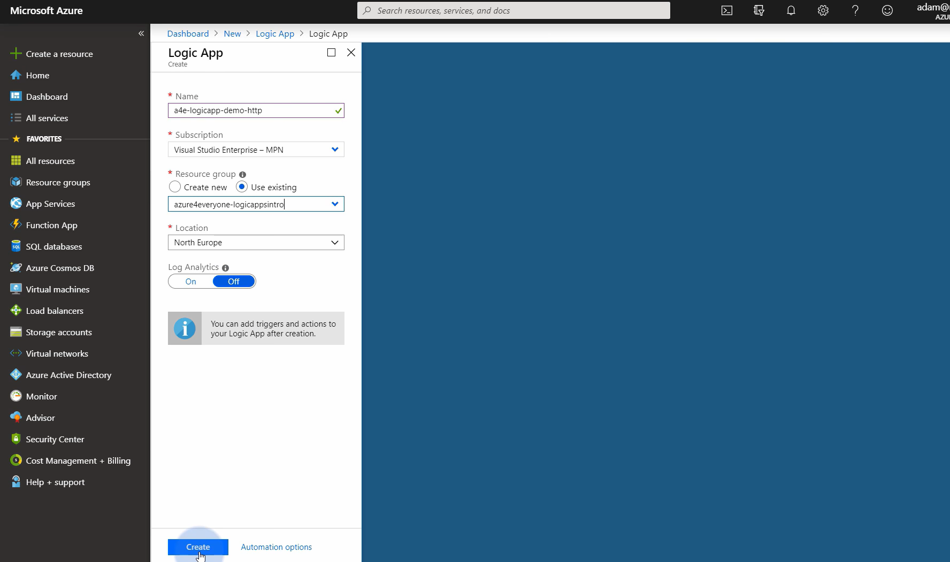


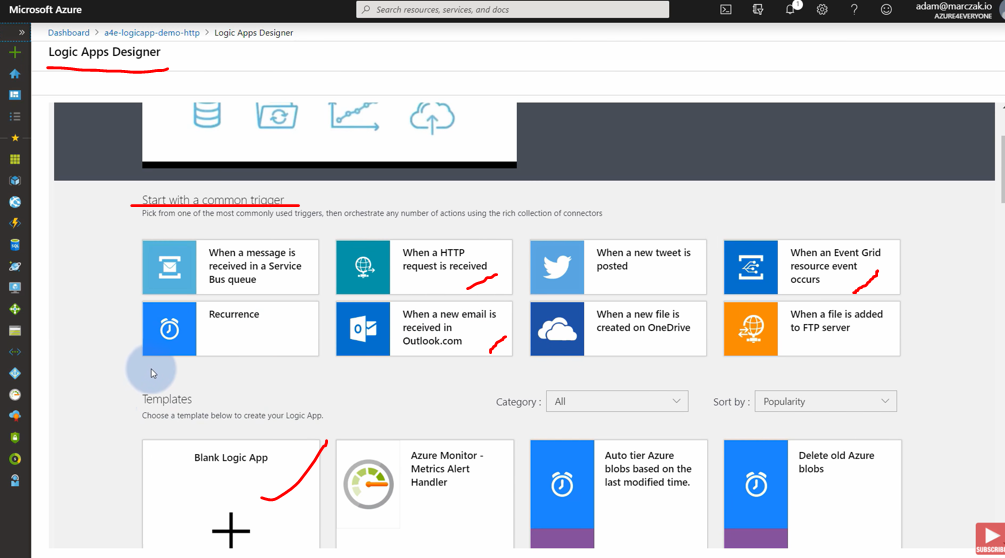


**First Demo -> Our first demo that's going to be happening today is this. We're going to have a user that's going to be shooting to the web request and going to get a simple response.**

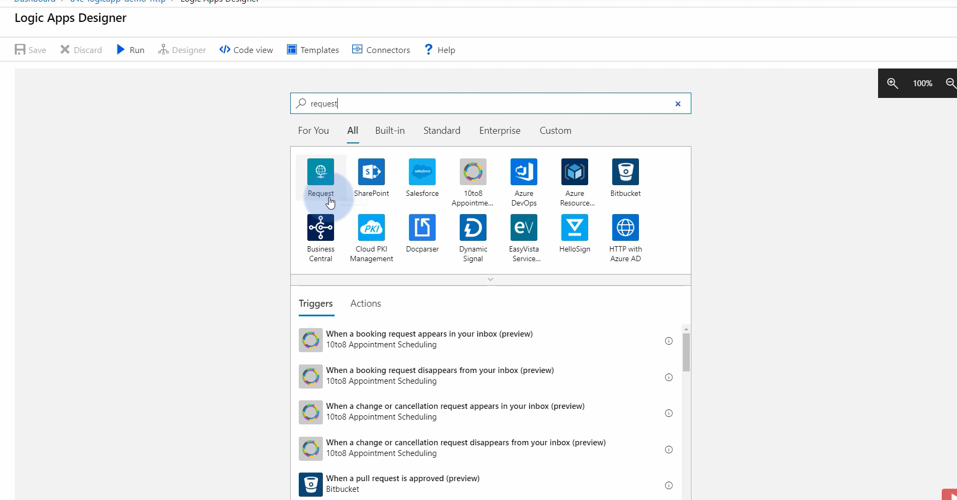


**Create Resource -> Logic App-> create**

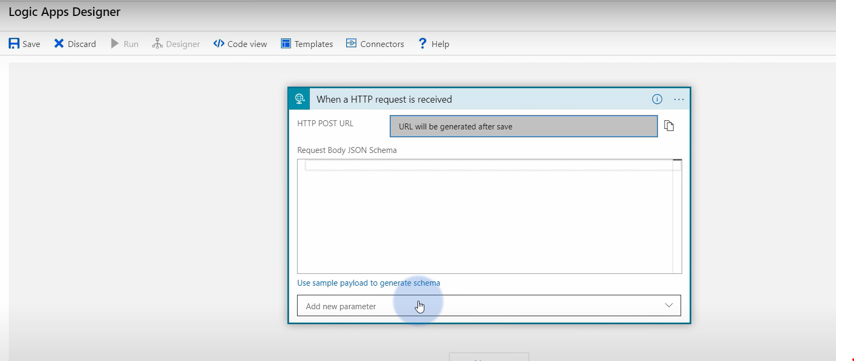




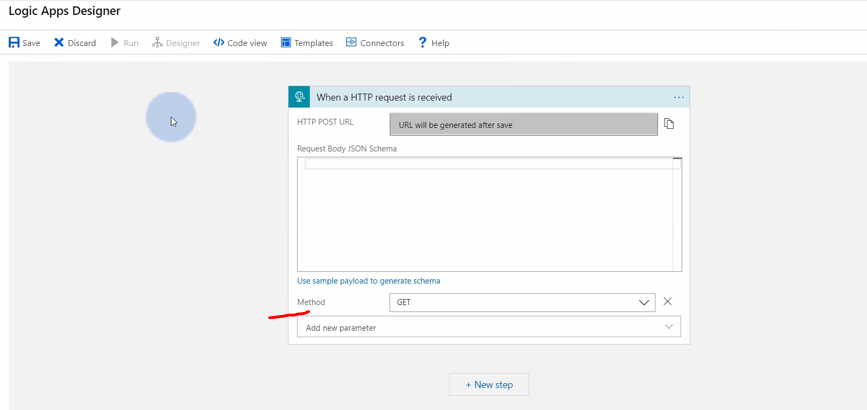
Go to Azure Portal. In order to create resource, we're going to hit create resource button here. Type a logic app. You need to select a subscription and a resource group. So I'm going to select the resource group that they created previously. you can go and hit go to resource. you are immediately moved to the logic app designer. This is your editor where you're going to be creating logic apps. So we are going to start with a blank logic app Template.

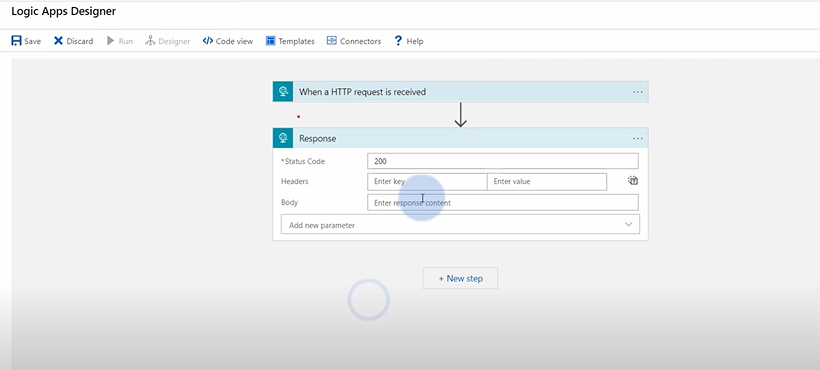


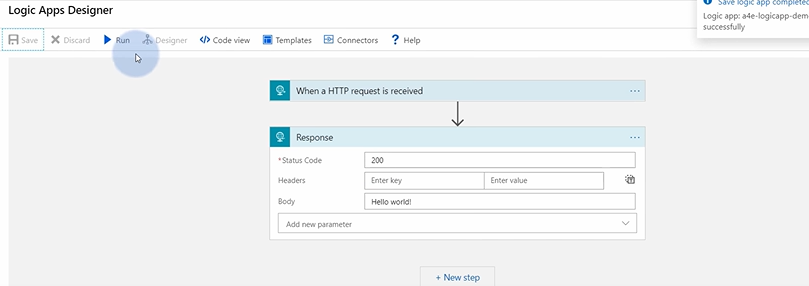
## 

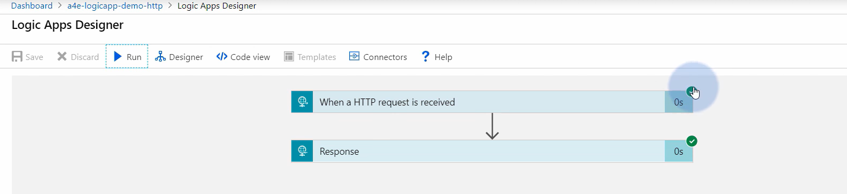


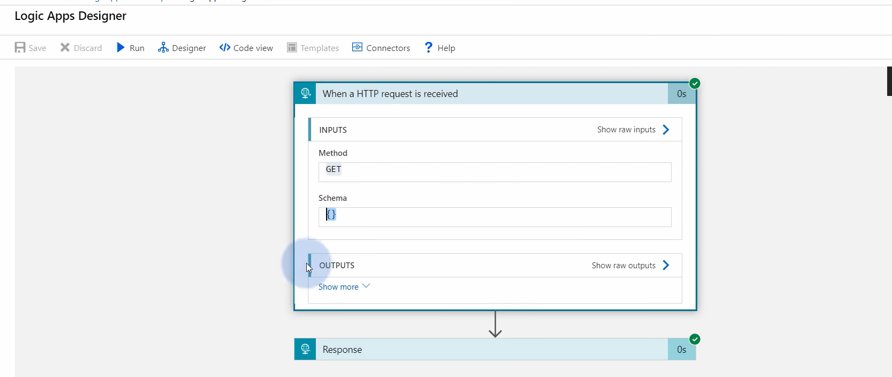
Since we want to create a web request, we just need to type request and select the first thing that pops up and now we need to specify what kind of request is this. So, there's going to be HTTP request that is being received. So, I'm going to select it. it says we're going to get a request URL after we save this. Now, even though it says HTTP post URL, you will be able to invoke this by get request and now, since this is configured, let's hide it and click new step. So, we now need to respond. So, let's type response. Notice on the below, we have a request response. This is an action that will respond to the user whenever the URL is hit. So, we want to respond with 200, which is success and maybe as a body, let's say hello world. and hit save. Our Logic App was saved. So, first of all, let's test it. You can always test it by using run button.

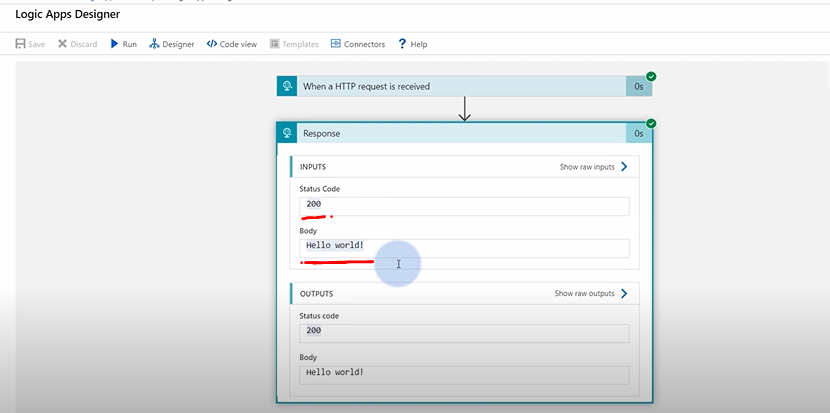




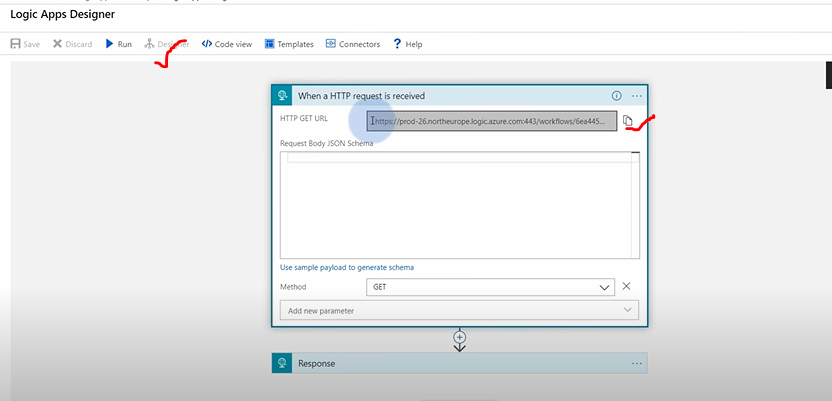




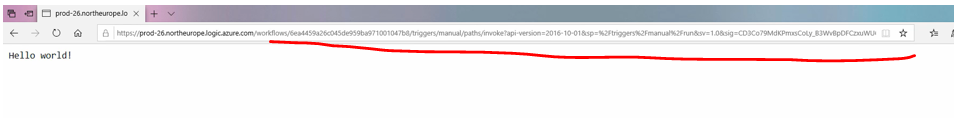




So, if this actually works, we can go back to designer, click on HTTP request and notice that this says now HTTP get URL, because we saved it. So, let's grab this URL, that you can pretty much copy into the browser. I'm just going to paste this URL here and hit enter.



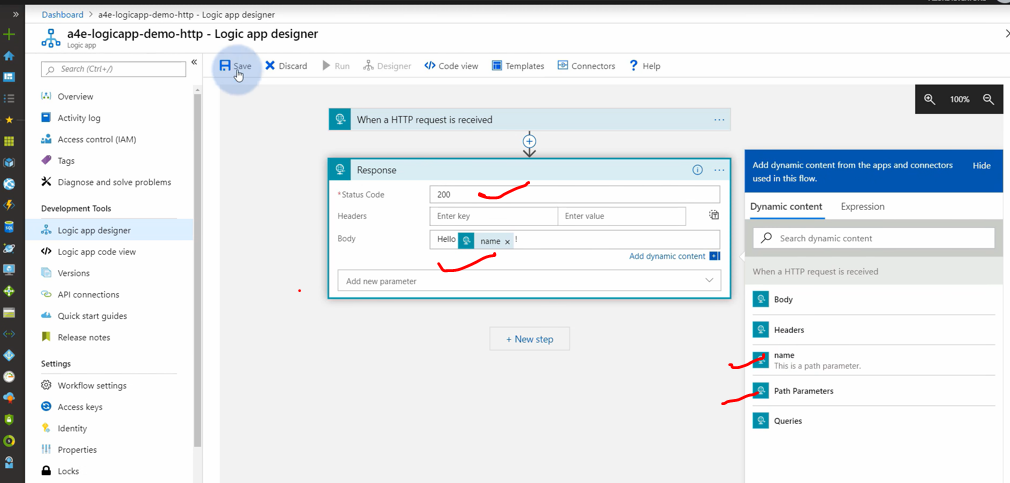
So, if this actually works, we can go back to designer, click on HTTP request and notice that this says now HTTP get URL, because we saved it. So, let's grab this URL, that you can pretty much copy into the browser. I'm just going to paste this URL here and hit enter. Notice that we got our response, hello world, which actually means everything is working just fine.





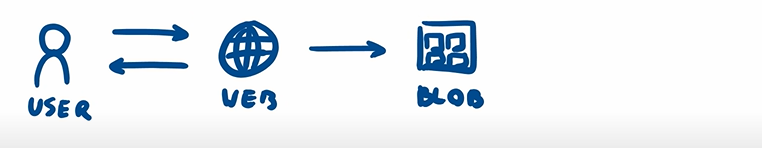


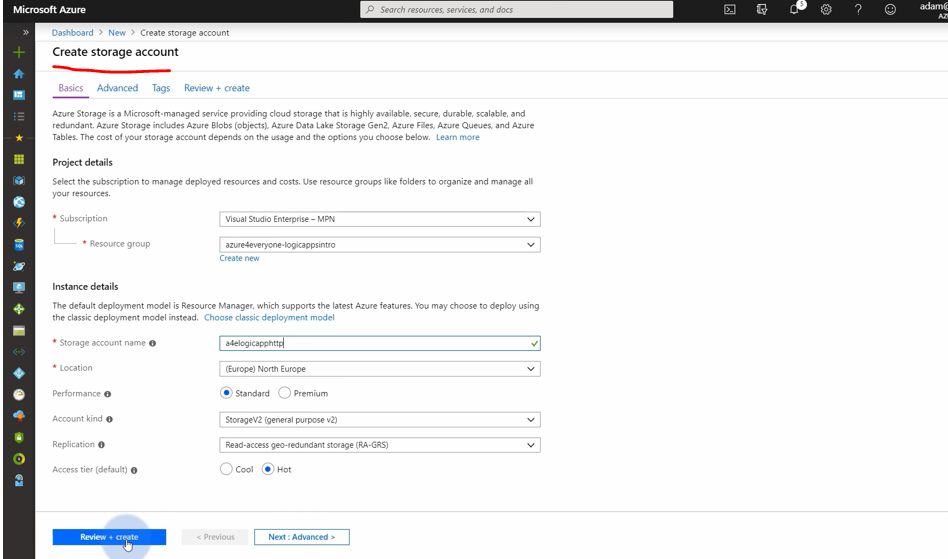
So, let's hit that URL again. Notice that our response now was Hello Adam. This is because of course we were able to pass and use a parameter in our flow.

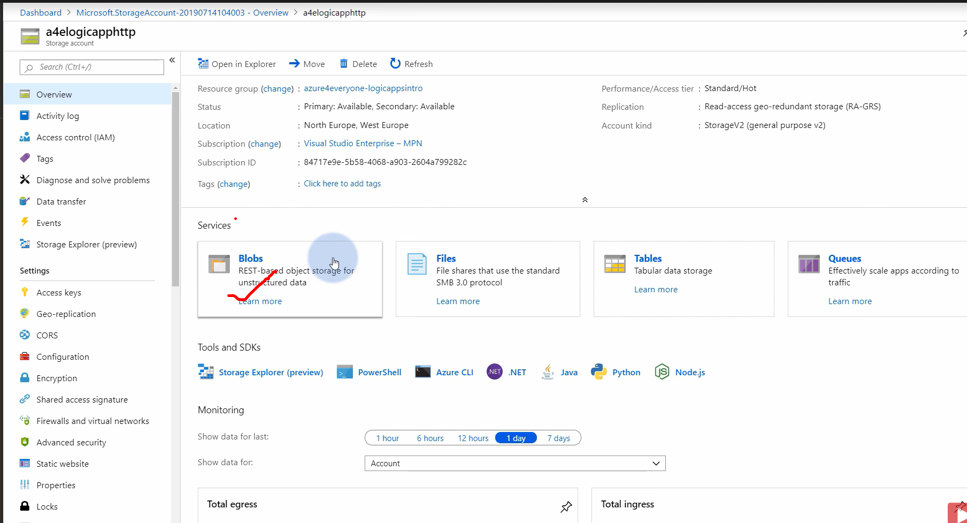


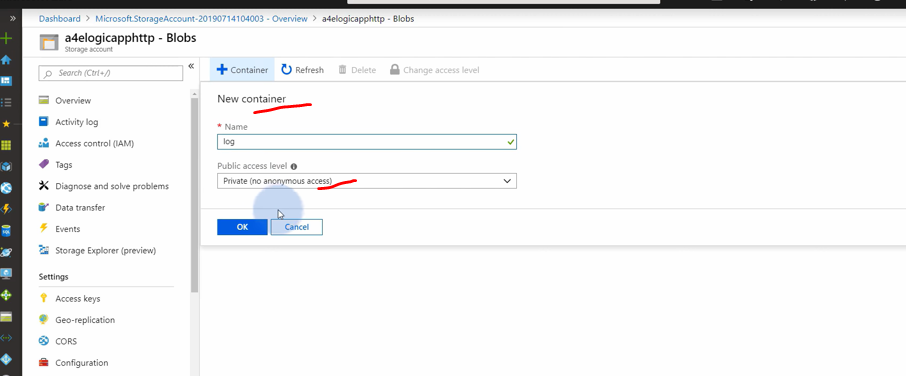


**Second Demo->** when we go back to response, maybe do something more complex. So, we just finished this example. Maybe we want to get a bit more complex. Let's maybe save every request that we created on a blob storage. Blob storage is a powerful service that allows you to store files.

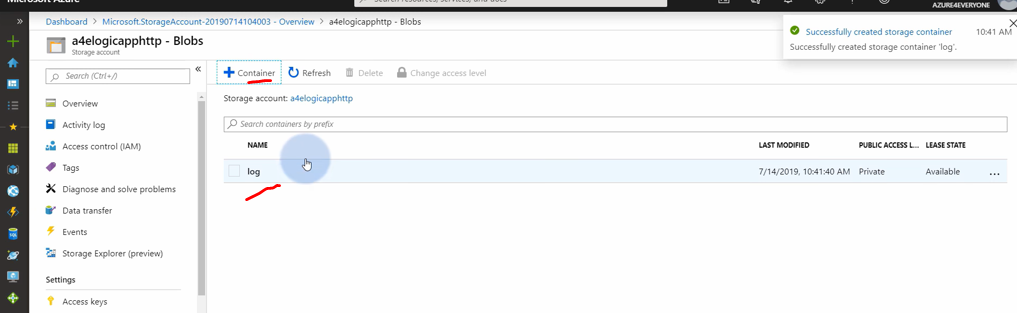


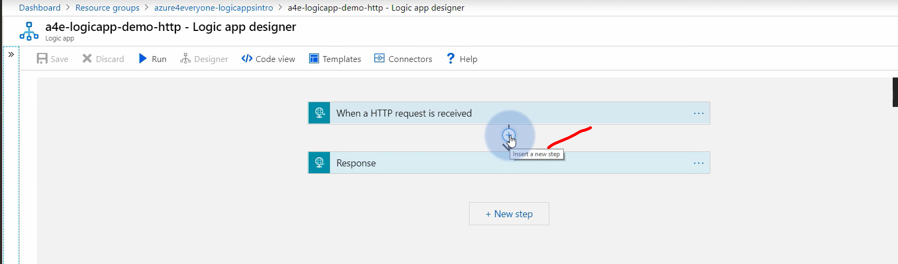




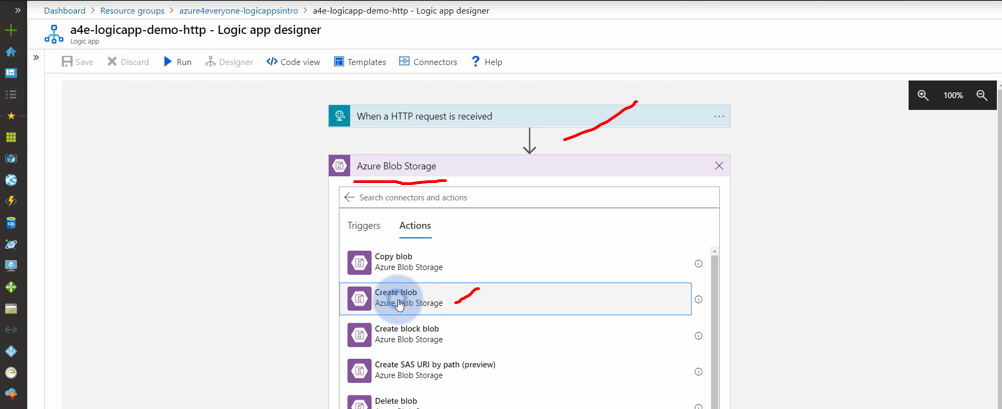


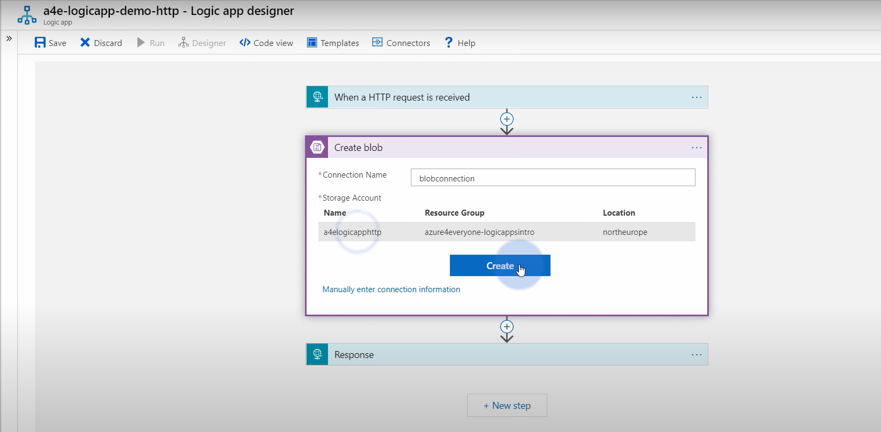
we don't have any blob storage yet, so we need to create it. So we hit create resource. We can type storage account, in order to create a storage account, you need to provide a resource group. We're going to create it in the same one that we already have. so, let's go to the resource, and this is our blob storage. We first always need to create a container, which is like a partition for our files. I'm going to create a container. It's going to be private because we don't want external people to access our files. I'm going to call it log. Basically, this is going to be our container containing all the files from our logic app execution. Since we have this set up, we can actually go back now to our logic app.

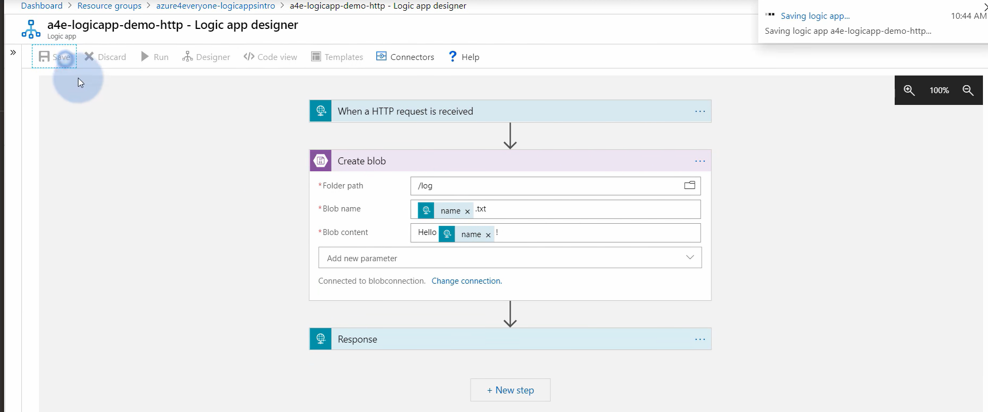




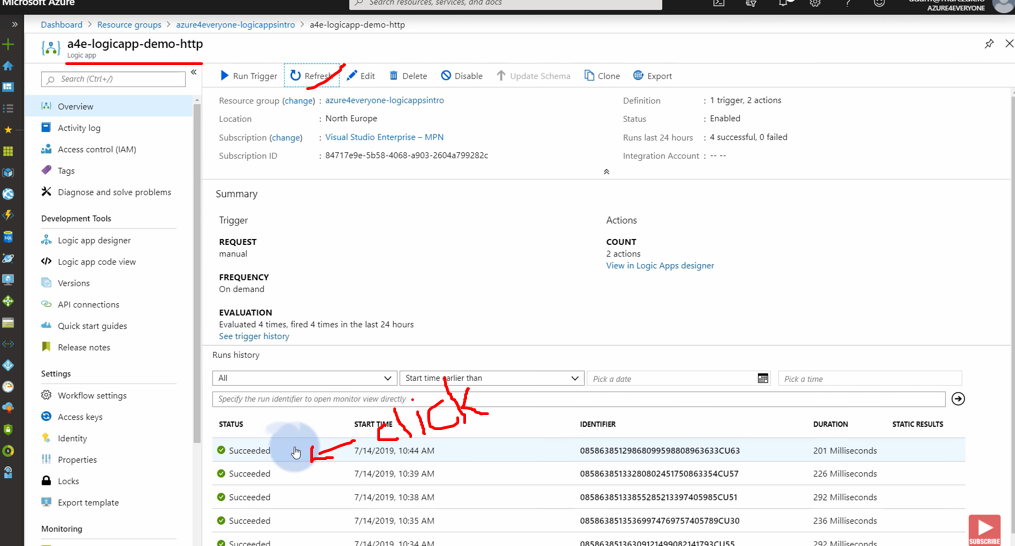
Now in the designer, let's go back to designer. Let's hit this button here, insert new step. This will allow us to add a step between a request and a response and add an action. Since we're going to be creating a blob storage integration, we just need to type a blob. First thing that will pop up is Azure Blob Storage. This is a connector that allows us to work with blob storage accounts without actually need of a technical knowledge. Let's hit it. We need to find an action that is called create blob because we're going to be creating new files. First thing that we need to define is a connection because our logic app is a public service, but our blob is a private service since we created a private container. It needs to know how to authorize and authenticate to this blob. We need to define a name. That's going to be our blob connection. We can either hit here because we already have access to our blob storage, or if this is external blob storage, you can enter connection information manually. For now, I'm just going to hit this and click create. What Azure does for us right now is connecting to this blob storage and grabbing blob storage name, URL, and an access key in order to create something called API connection. This is a safe connection information which allows logic apps to connect to blob storage in the future. Our connection just got saved. Since we want to create a blob, first of all, we need to select a folder. A folder is pretty much a container and then a path to the file. Let's hit it and select a log. This was our container that we just created. We need to specify the blob name. Maybe I'm just going to use a name, but we need an extension since those are files. I'm just going to add txt. We need also some sort of content, so let's just type hello world and a name. Instead of world, let's use a parameter. Notice that it says we don't have any parameters, but it's just hidden, so we need to hit show more, click name. Now, if this works, this means when we hit save and we go back to our URL and run it again, that means if we go back to logic app, we can of course validate if this runs successfully by looking at the log.





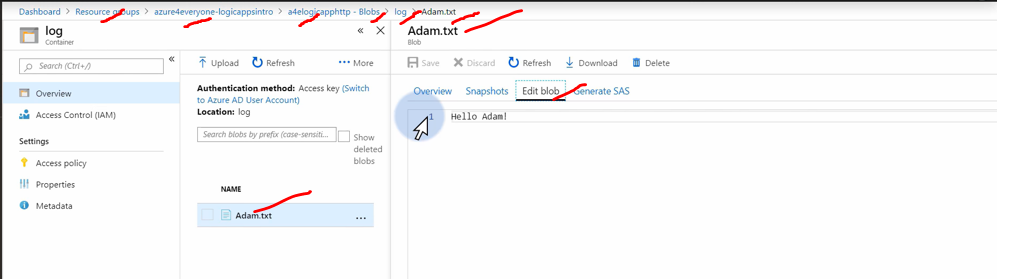


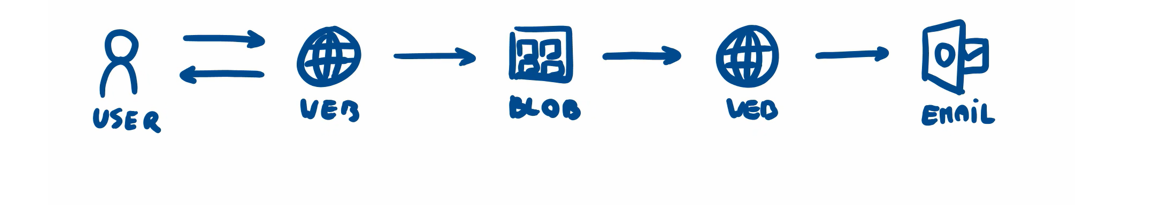


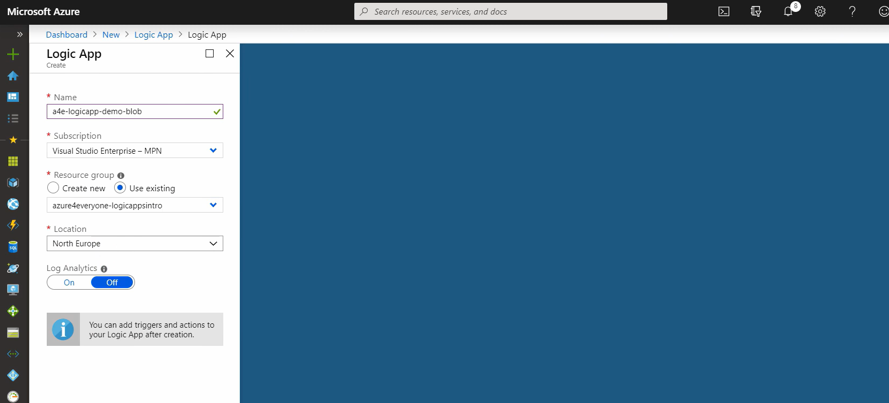


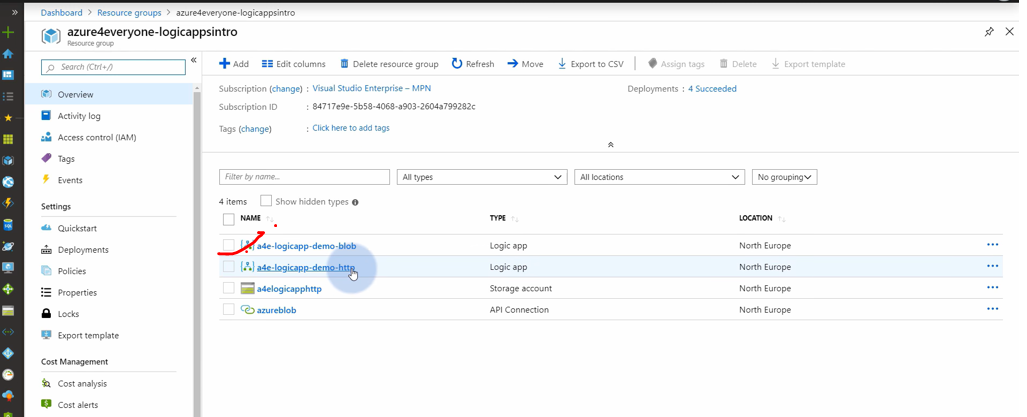


If this actually worked, we can go to our resource group and go back to our storage account, go to blob, log container, and see, this is our file, adam.txt. In order to validate it actually got created correctly, you can always also click on it, go to edit blob, and see the contents. This is great. Let's go back to our logic app. Let's do something a bit more complex.





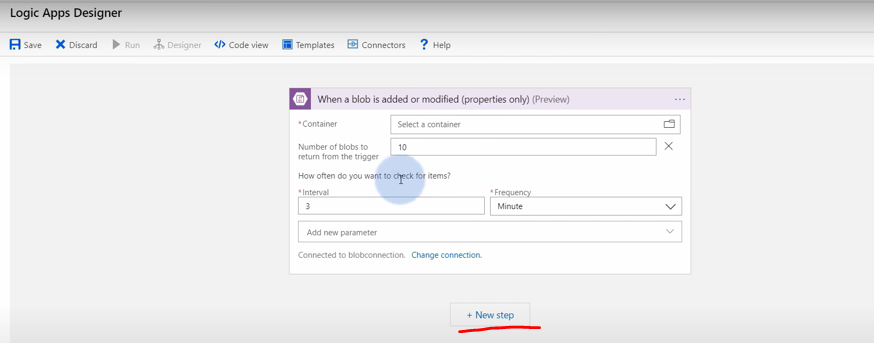


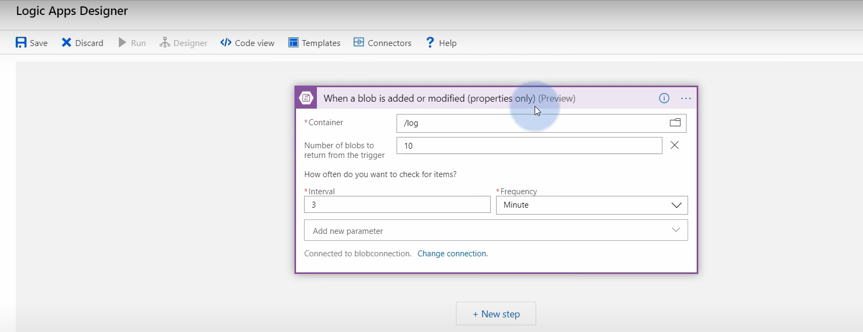


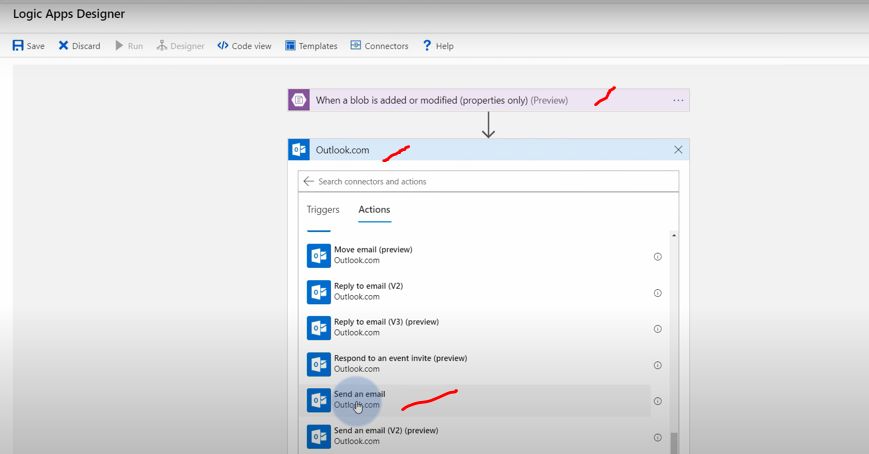
**Third Demo** let's create another scenario where logic app reacts to the new files on the blob, and after new files appear, it will send us an email. Maybe let's do that with an outlook. In order to do that, we will need another logic app, because this one already has a trigger, and it's a HTTP trigger. Right now, we will need a trigger that reacts whenever there's a new blob. To do that, we go to new resource, type in again logic app, hit create.

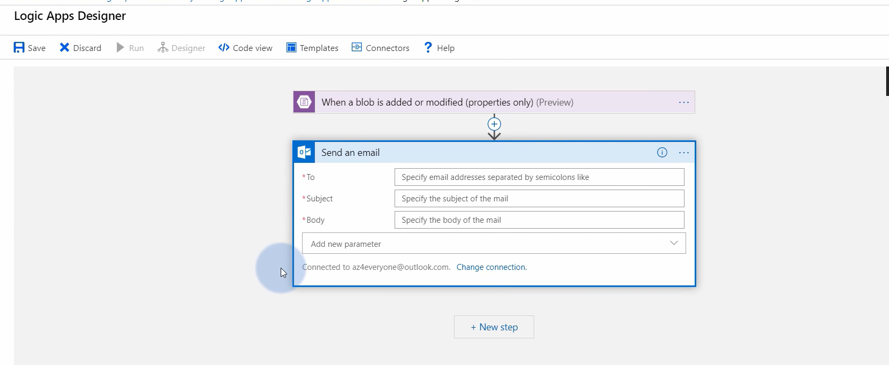
Notice that because I actually kept the naming convention, they're nicely aligned, and I know very quickly that this is a blob, and this is HTTP logic app. Go to blob storage. As usual, you're prompted with an empty designer, so let's start with a blank logic app, and let's type blob. As you see, we don't need to type blob, because we already worked with the blob connectivity, and Azure remembers that through the cookies. We can actually hit Azure blob storage here, and there's only one trigger, which is called when the new blob is added or modified. Let's use that.

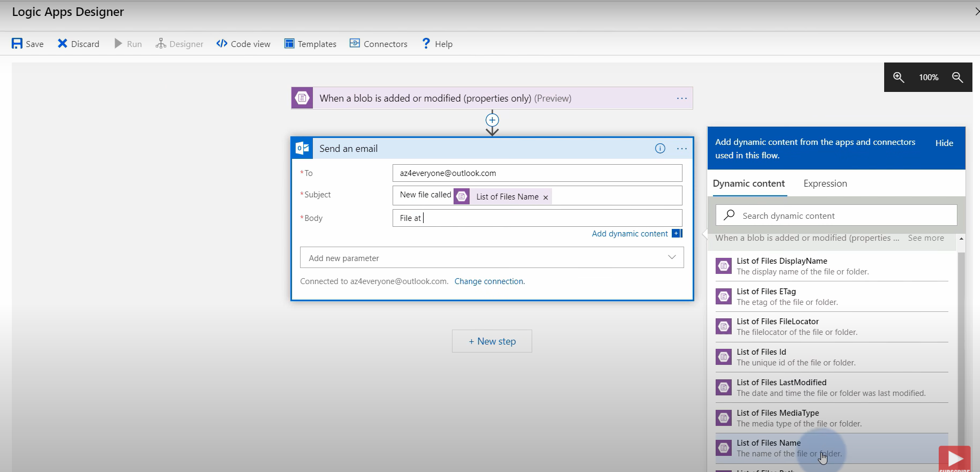


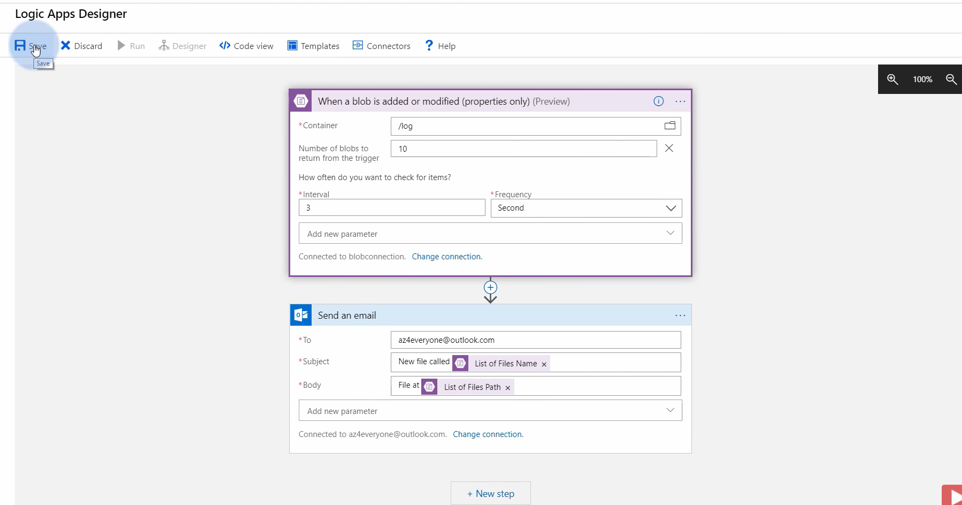




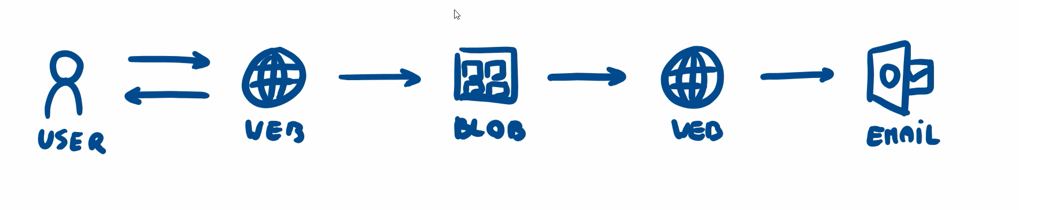


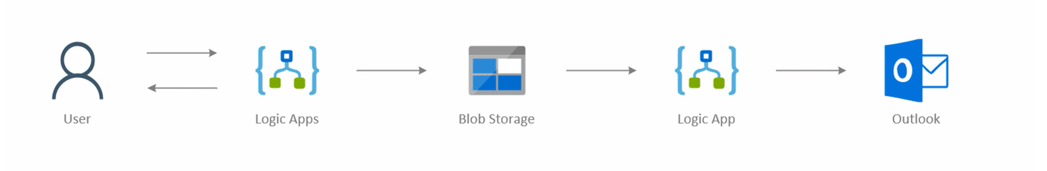






if this Logic Apps right now works, notice that we already are getting emails. We got a new file called adam.txt file that was saved at log slash adam.txt. So, we created also John and Tom. See, we already got them. But in case something would not work, you can always go back to your Logic App, refresh here, and check the execution if you missed some steps. It's a powerful feature that will also tell you if you missed something up. Let's go back to our diagram. As you see, we are able to implement something called function chaining using Logic Apps. It looks even better as a nice diagram using Visio. And that's pretty much it.





**Important Links:**

[**https://www.youtube.com/watch?v=inaXkN2UrFE&list=PLGjZwEtPN7j-Q59JYso3L4\_yoCjj2syrM&index=10**](https://www.youtube.com/watch?v=inaXkN2UrFE&list=PLGjZwEtPN7j-Q59JYso3L4_yoCjj2syrM&index=10)

**Azure Logic App**

<https://www.youtube.com/watch?v=ZvsOzji_8ow>

**Azure App Service(Web Apps)**

<https://www.youtube.com/watch?v=4BwyqmRTrx8>

**Azure Function Apps( Introduction for Serverless programming)**

<https://www.youtube.com/watch?v=Vxf-rOEO1q4>

**Azure Cosmos DB(Globally distributed NoSQL database)**

<https://www.youtube.com/watch?v=R_Fi59j6BMo>

**Azure SQL database(Relational Database in Azure)**

<https://www.youtube.com/watch?v=BgvEOkcR0Wk>

**Azure Event Grid**

<https://www.youtube.com/watch?v=TujzkSxJzIA>

**Azure Container Instances( Serverless containers in cloud)**

<https://www.youtube.com/watch?v=jAWLQFi4USk>

**Azure Storage| Introduction to Blob, Queue, Table & File Share**

<https://www.youtube.com/watch?v=UzTtastcBsk>

**Integration**

**Azure App Service**

[**https://www.youtube.com/watch?v=NNjgt\_8w9V4**](https://www.youtube.com/watch?v=NNjgt_8w9V4)

**Microsoft AzureSQL**

[**https://www.youtube.com/watch?v=puVMCRvdE9A**](https://www.youtube.com/watch?v=puVMCRvdE9A)

azuresingh263@gmail.com

**Azure In Deep Dive Link-**

**Azure -> https://portal.azure.com/**

**Microsoft Azure Fundamentals (AZ-900)**

[**https://www.youtube.com/watch?v=inaXkN2UrFE&list=PLGjZwEtPN7j-Q59JYso3L4\_yoCjj2syrM&index=10**](https://www.youtube.com/watch?v=inaXkN2UrFE&list=PLGjZwEtPN7j-Q59JYso3L4_yoCjj2syrM&index=10)

**Azure Logic Apps**

[**https://www.youtube.com/watch?v=ZvsOzji\_8ow&list=PLGjZwEtPN7j\_nFYA1qc7cWumNr89PkfAX**](https://www.youtube.com/watch?v=ZvsOzji_8ow&list=PLGjZwEtPN7j_nFYA1qc7cWumNr89PkfAX)

**Azure Storage**

[**https://www.youtube.com/watch?v=UzTtastcBsk&list=PLGjZwEtPN7j9hIHIQJ1Uh6IxEImhWZkPy**](https://www.youtube.com/watch?v=UzTtastcBsk&list=PLGjZwEtPN7j9hIHIQJ1Uh6IxEImhWZkPy)

**Azure Storage Account**

[**https://www.youtube.com/watch?v=UzTtastcBsk&list=PLGjZwEtPN7j9hIHIQJ1Uh6IxEImhWZkPy**](https://www.youtube.com/watch?v=UzTtastcBsk&list=PLGjZwEtPN7j9hIHIQJ1Uh6IxEImhWZkPy)

# Group coordinator | Group leader | Rebalance Internals | Apache Kafka Tutorial in Hindi | Video 12

<https://www.youtube.com/watch?v=n5kxkFNTZZ0>

# Consumer group in kafka | consumer group rebalancing | Apache Kafka Tutorial in Hindi | Video 11

<https://www.youtube.com/watch?v=zVSaT0glaX8>