APACHE SLING & FRIENDS TECH MEETUP

2 - 4 SEPTEMBER 2019

CIF the modern way to integrate commerce engines with Adobe Experience Cloud

Carlos Duque & Francisco Madeira, diconium

Carlos Duque

Francisco Madeira

adaptTo()

// diconium

BUSINESS Solution Partner

HYBRIS PLATINUM REGIONAL PARTNER

FOUNDED

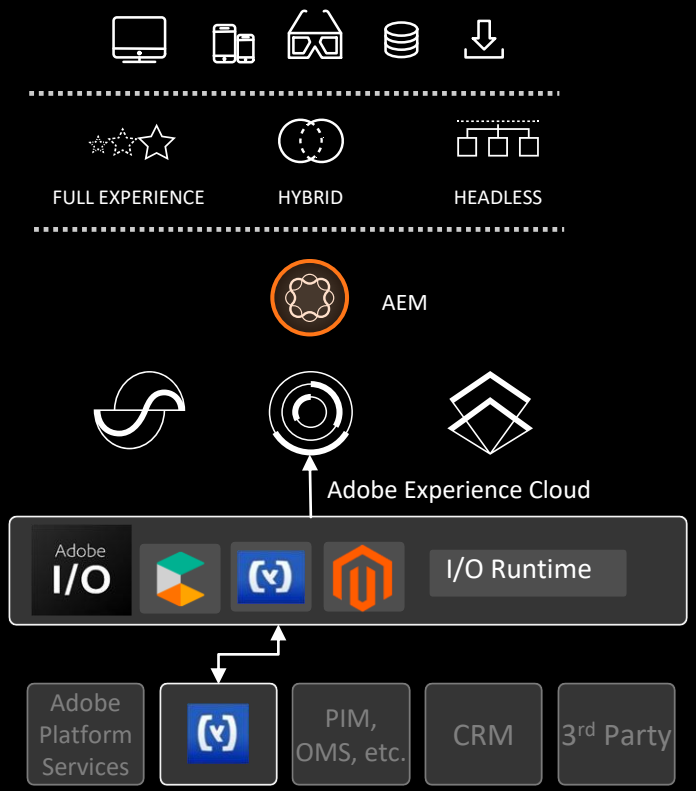
1995

EMPLOYEES

900

Locations: Stuttgart | Berlin | Hamburg | Karlsruhe | Bangalore | Detroit | Lisbon | London | Beijing | San Jose | U

Adobe Experience Cloud



- Deliver a common API across all Experience Cloud
  - Easily present commerce information using Adobe Experience Manager
  - Currently Available for **SAP CX Commerce**, Magento and Commerce Tools

adaptTo()

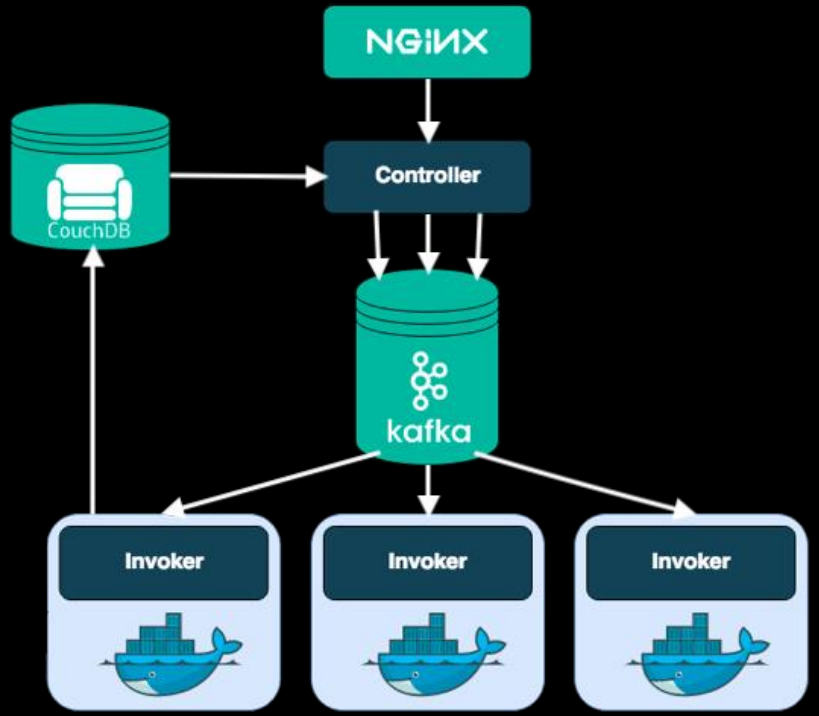
CIF – Commerce Integration Framework

“The CIF REST API is the heart of all programmatic interactions with Adobe's commerce integrations.“

- Open source
- Easily extendable
- CIF REST API can:
  - Retrieve product information
  - Search and filter products
  - Create omni channel checkout experiences

Adobe I/O Runtime

- Serverless
- Apache OpenWhisk
- Microservices
- Auto Scale



Adobe I/O Runtime

- Cluster deployed in both the US and Europe for fast performance everywhere
- Theoretical infinity number of resources since it’s a serverless platform
- Limitations will be present in the commerce platform
- It provides a caching mechanism for better performance

Adobe I/O Runtime - Development

- Development is done using node.js (other programming languages supported by openwhisk will soon be available)
- Developers are focused on the code and not the infrastructure

Adobe I/O Runtime - Development

- No local environment
- Needs a very good unit testing suite – only way to test before deployment
- Very good de CI/CD process for easy and fast deployment

DEMO

SAP CX Commerce

- Formerly known as SAP Hybris
- Commerce Engine
- Provides out of the box REST interface
- Categories, Products, Carts, Orders, Users

diconium/commerce-cif-hybris

https://github.com/diconium/commerce-cif-hybris

diconium/commerce-cif-hybris

- Available on GitHub
- Developed in TypeScript
- Lerna for monorepo, one package for each commerce entity
- Build with yarn, webpack and serverless framework
- Mocha and Chai for building a solid test suit
- Deployed with circle ci

adaptTo()

diconium/commerce-cif-hybris

- The first project using the connector is close to production
- We've found it easier to implement since it was possible to start with the OOB features and extend it from there
- A second one it's on the way

diconium/commerce-cif-hybris

- Each endpoint is a sequence of actions:
  - Validation
  - Service Call(s)
  - Web Action Transform

diconium/commerce-cif-hybris

- Get Product Serverless declaration:

```
getProductById:
    name: ${self:custom.customer-package}/getProductById
    sequence:
      - ${self:custom.productsactions}/validateGetProductService
      - ${self:custom.products-actions}/getProductService
      - ${self:custom.common-actions}/webActionTransformer
    annotations:
      web-export: true
```

diconium/commerce-cif-hybris

- Get Product action:

```
function getProductById(input: Input): Promise<Output<Product>> {
    return new SimpleAction<Product>(input)
            .setMapper(ProductMapper)
            .setClient(GetProductByIdClient)
            .setErrorType(ERROR_TYPE)
            .activate();
 }
```

DEMO

adaptTo()

AEM Advantages

- Faster time to market
  - Reuse pre built frontend components
  - OOTB product catalog configuration
  - Less development times
  - Focus on end user experience
  - Works with MPA and SPA

DEMO

AEM Development

- Use modern presentation tools like:
  - Web Components 
  - React 
  - Angular 

AEM Development

- HTL:

```
<sly data-sly-test="${!properties.productId}">
    Choose a product from dialog.
</sly>

<sly data-sly-test="${properties.productId}">
    <product-picker-component
            data-sly-attribute.data-product-id="${properties.productId}”>
    </product-picker-component>
</sly>
```

AEM Development

- Web Component:

```
class ProductModel extends DOMModel {
  @byAttrVal('data-product-id') productId;
}
const ProductPickerWebComponent = createCustomElement(
  ProductPickerComponent,
  ProductModel,
  'element',
);
if (!window.customElements.get(ProductPickerWebComponent.elementName)) {
  window.customElements.define('product-picker-component',
    ProductPickerWebComponent,
  );
}
```

AEM Development

- React Component:

```
export const ProductPicker = ({ productId }) => {

  const product = useProductGet(productId);

  return (
      . . .
  );

};
```

DEMO

Next Steps

- Open sourcing of a sample project
- Add support for GraphQL

QUESTIONS?

adaptTo()

Thank You ☺

adaptTo()

References

- https://github.com/diconium/commerce-cif-hybris
- https://www.adobe.io/apis/experienceplatform/runtime.html
- https://www.adobe.io/apis/experiencecloud/commerce-integration-framework.html
- https://medium.com/adobetech/introducing-adobe-i-o-runtime-how-serverless-at-adobe-will-shape-the-future-of-digital-marketing-56ad60852477
- https://diconium.com/en/news/adobe-io-hybris
- https://opensource.adobe.com/commerce-cif-api/?urls.primaryName=CIF%20Cloud%20API%201.1.2#/
- https://github.com/adobe/commerce-cif-api
- http://openwhisk.apache.org/
- https://www.webcomponents.org/
- https://github.com/adobe/react-webcomponent