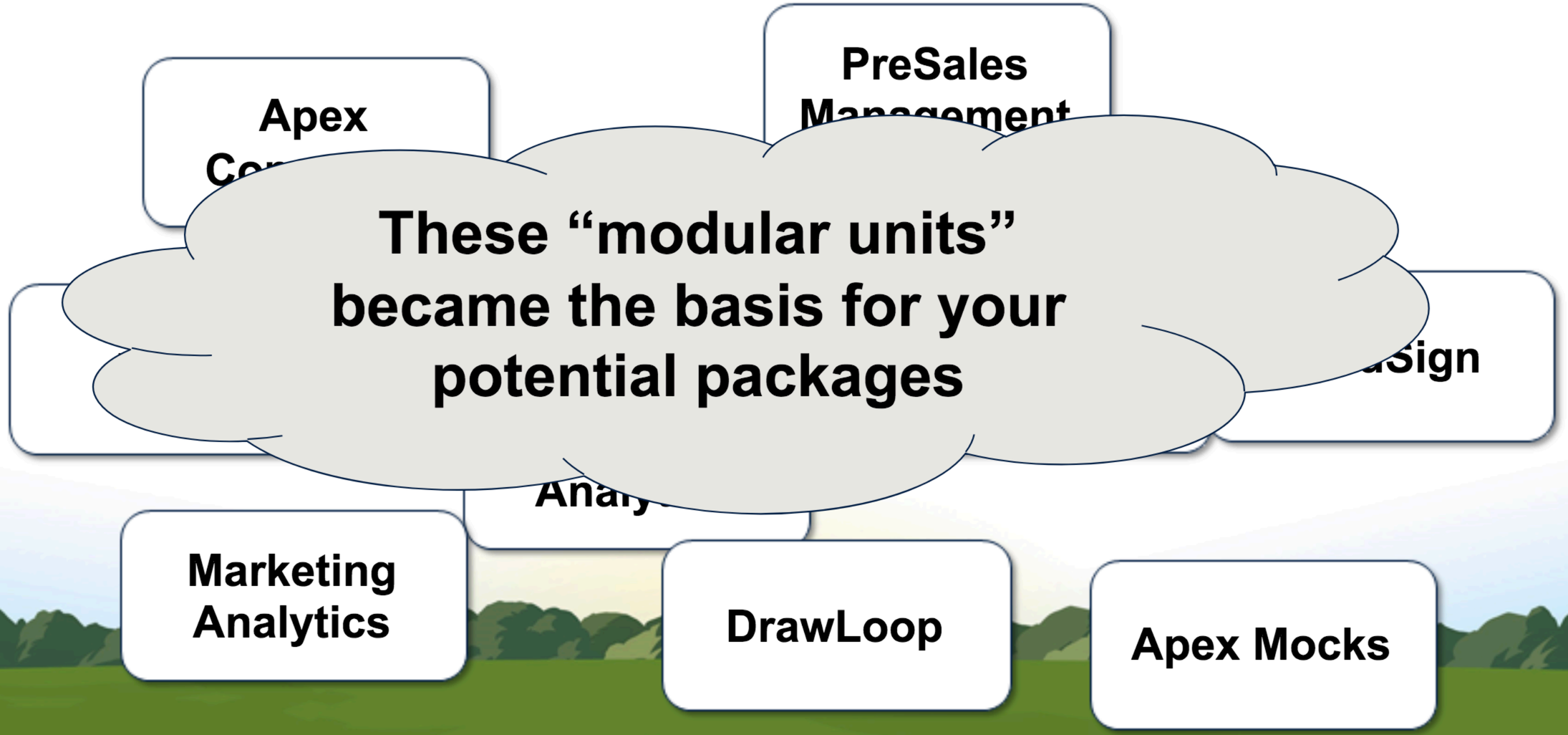
Packages

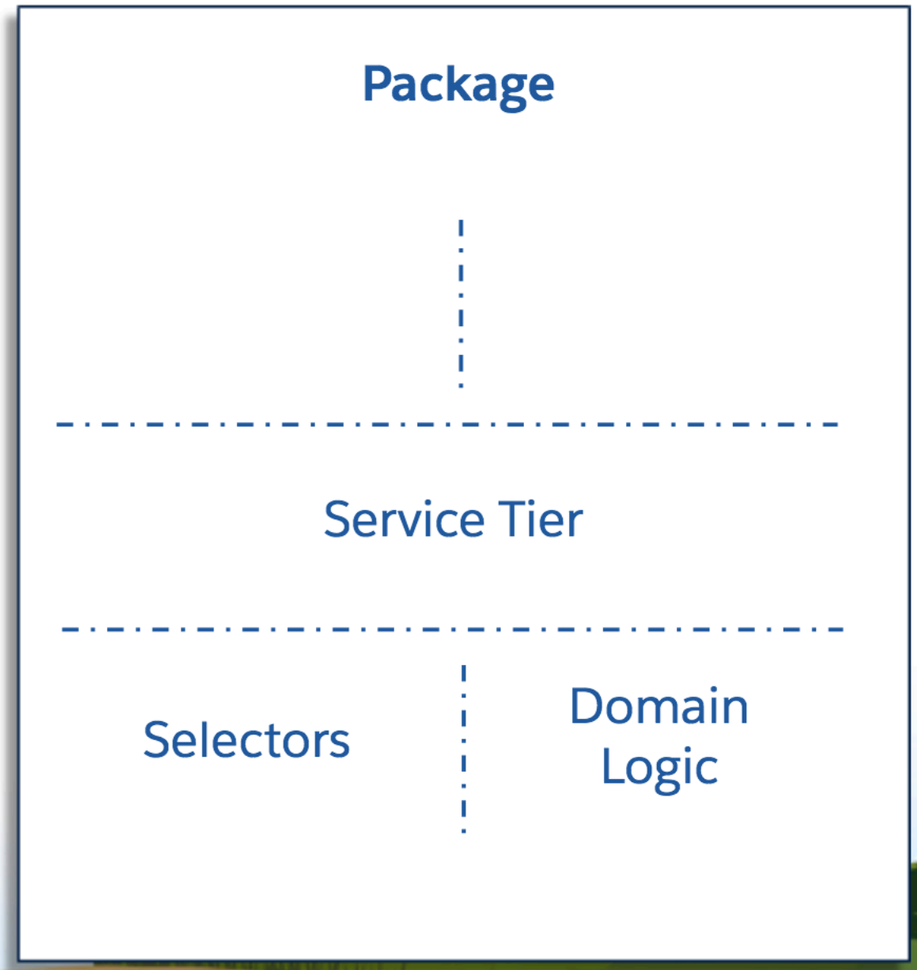
Pretty soon, things just start to become clear how they will be packaged



Organize Your Metadata By “Purpose”

Refactor each module by its purpose.

- Selector classes
- Domain Logic classes
- Service Tier classes



Adopt Naming Conventions

Strict naming convention to help keep things organized

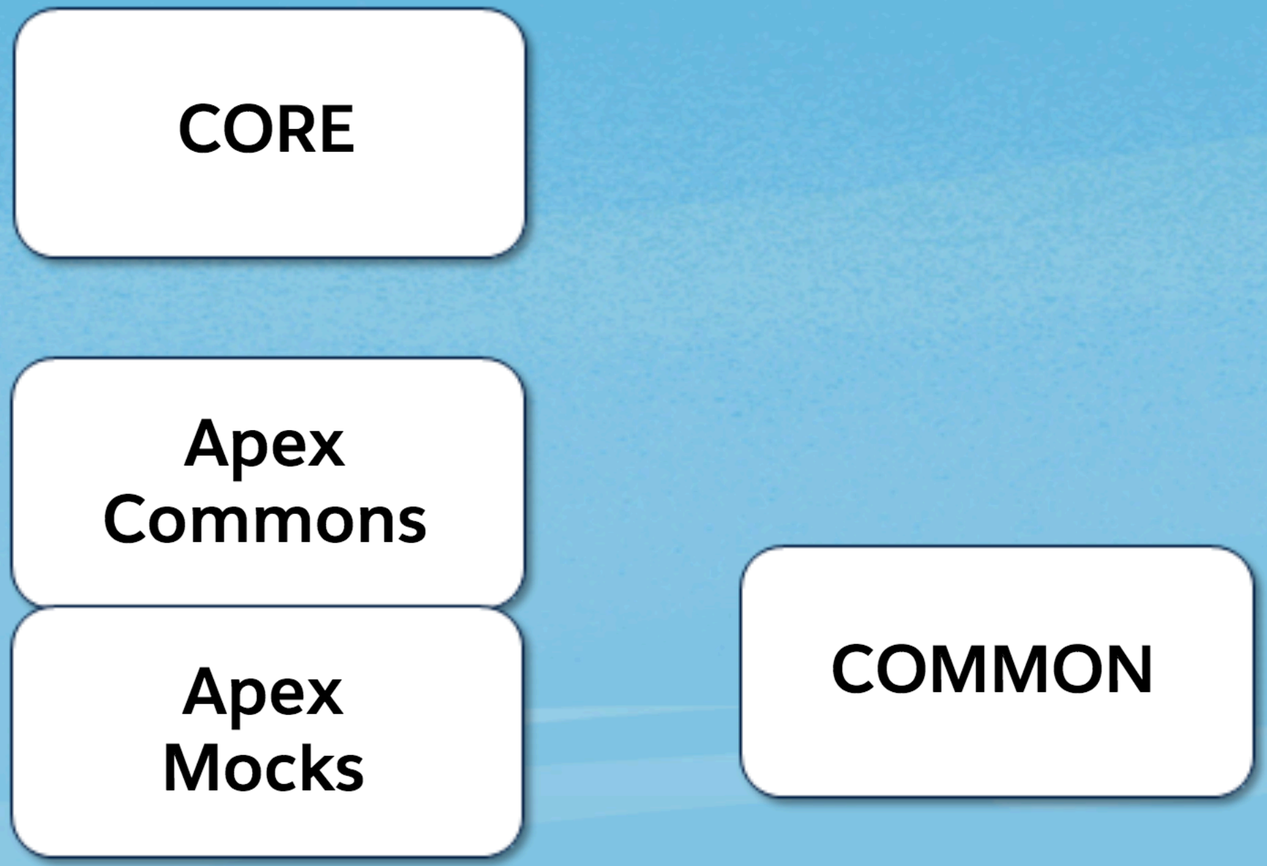
Built around two specific ideas:

“Package” Prefix **“Meaningful Descriptor” Suffix**

common _AccountsSelector

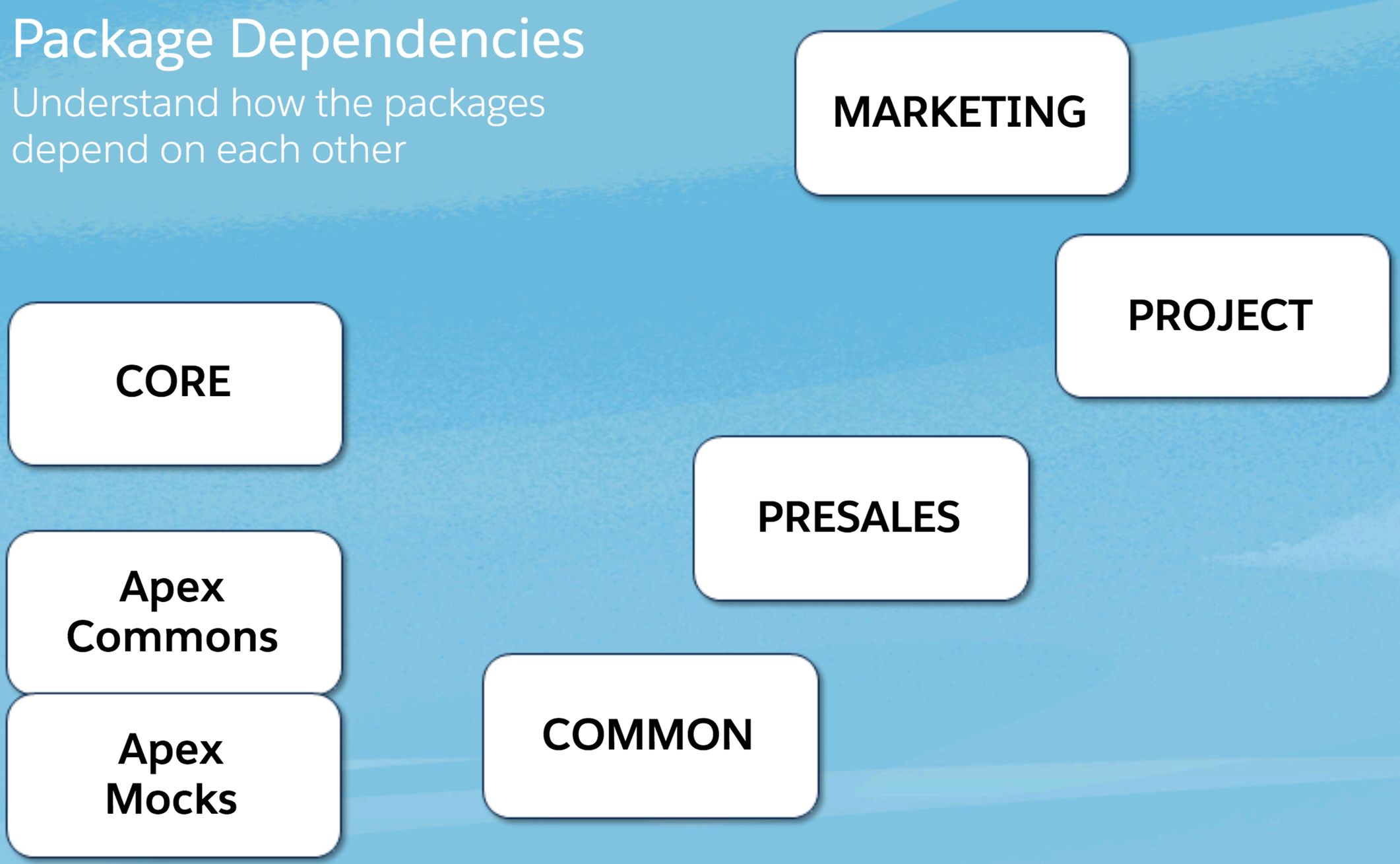
Package Dependencies

Understand how the packages depend on each other



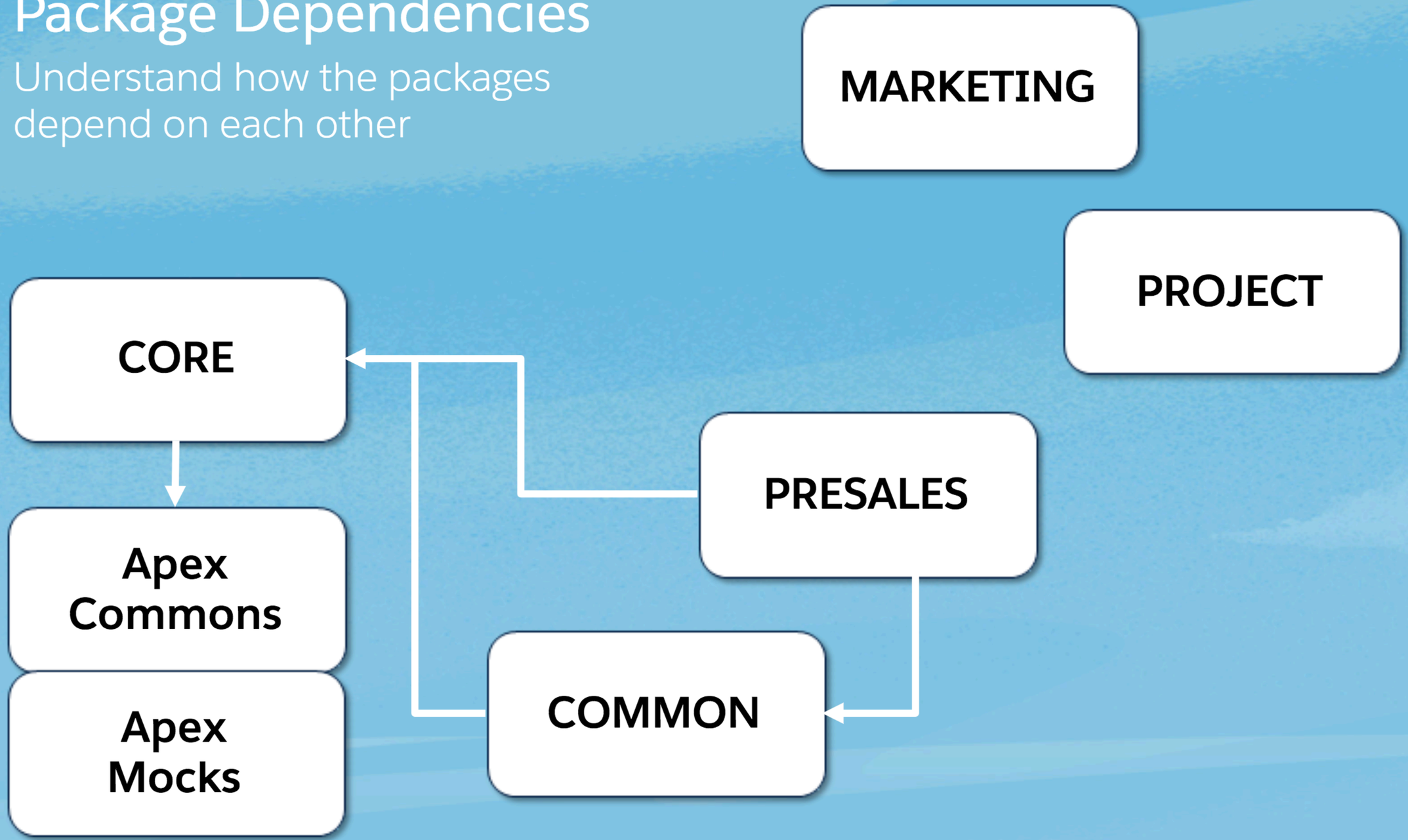
Package Dependencies

Understand how the packages depend on each other



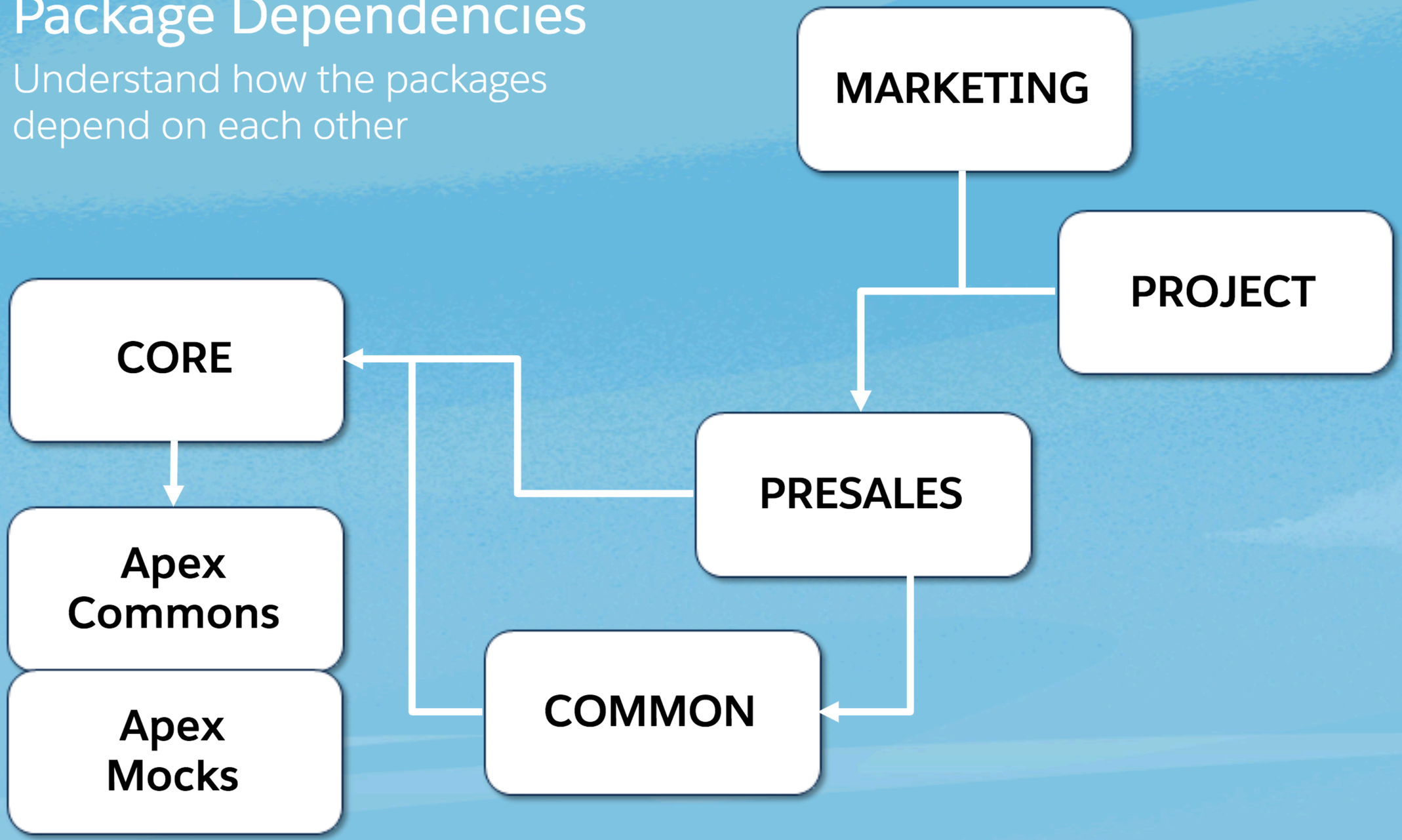
Package Dependencies

Understand how the packages depend on each other



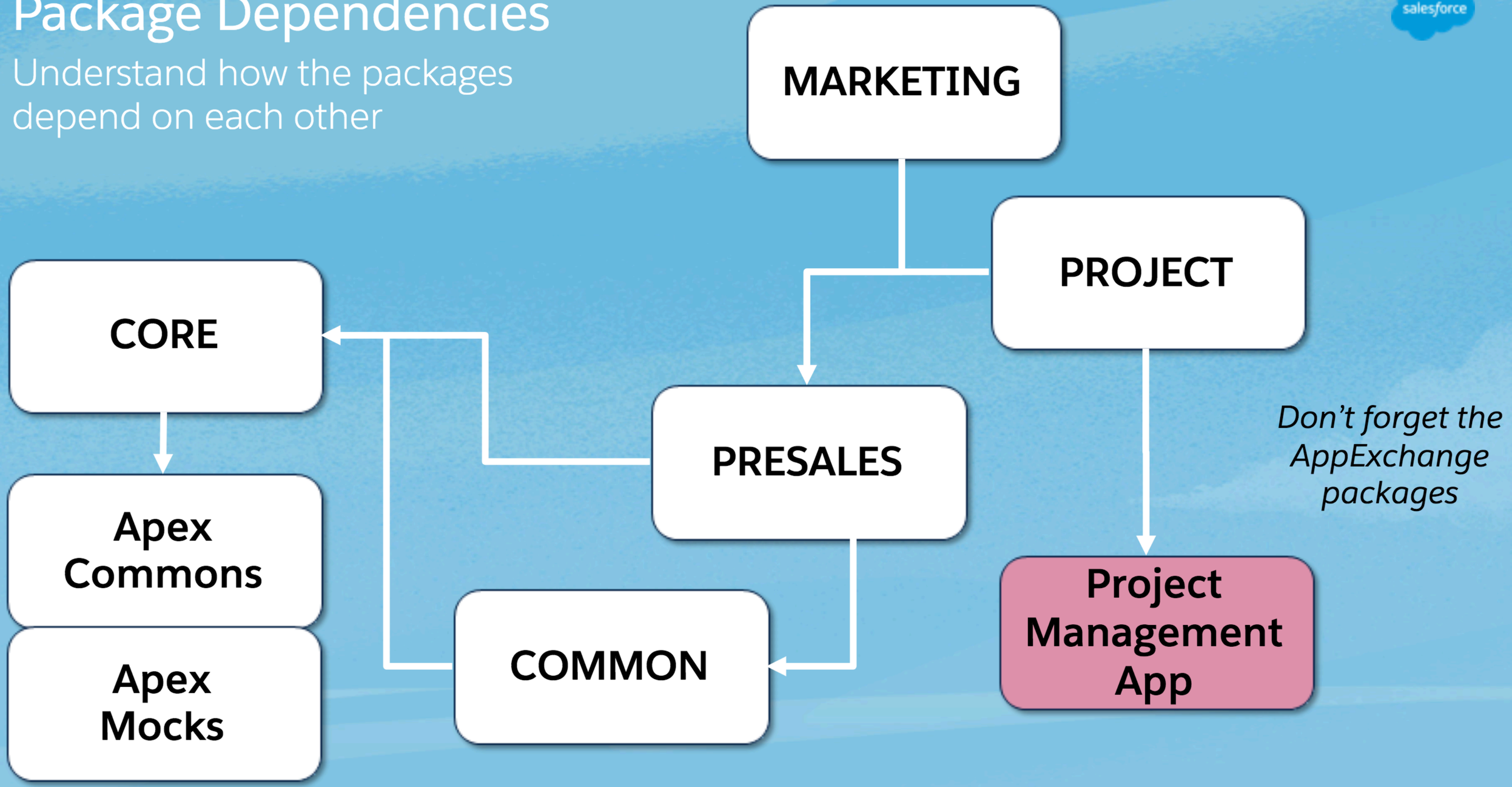
Package Dependencies

Understand how the packages depend on each other

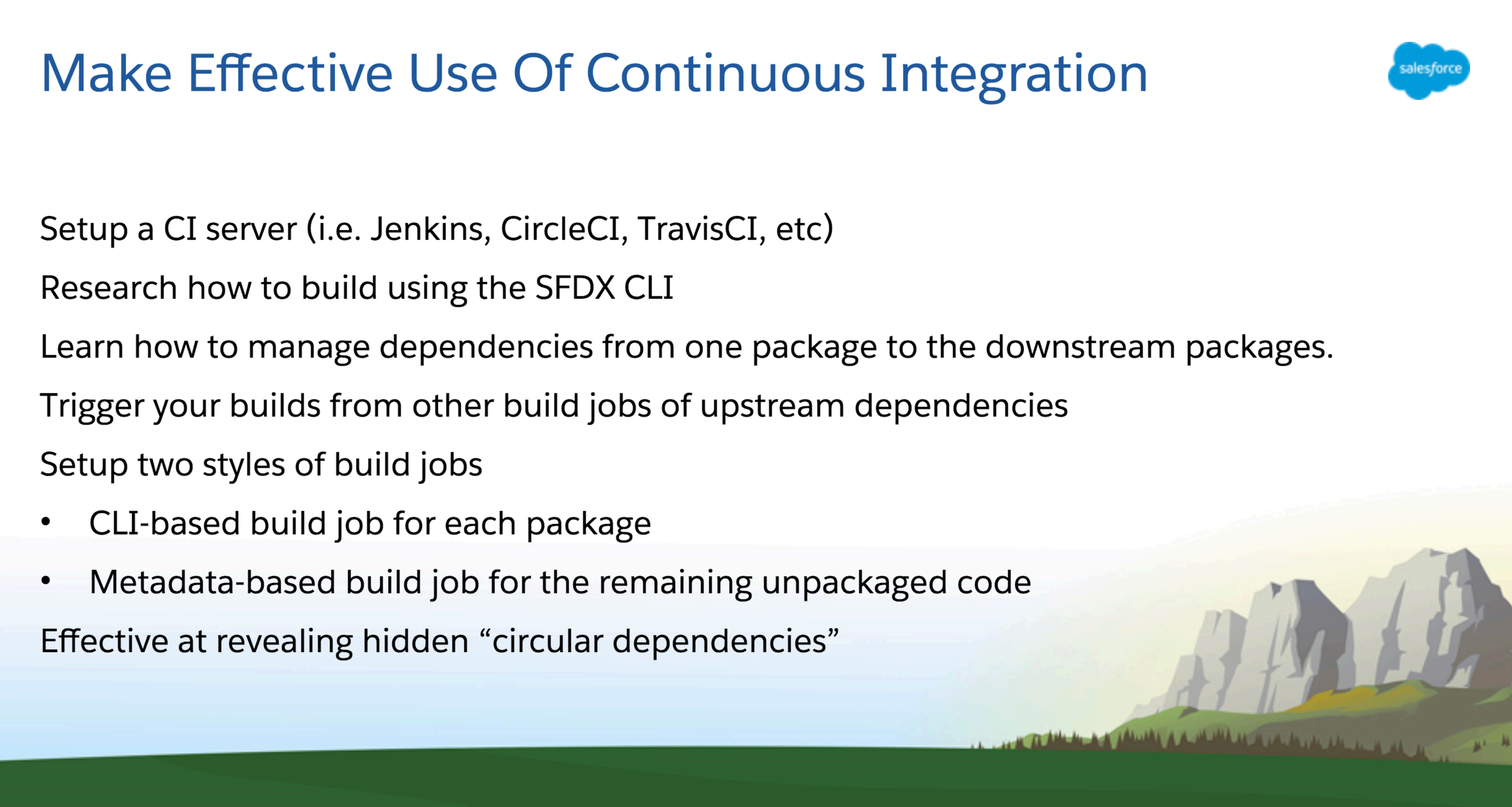


Package Dependencies

Understand how the packages depend on each other



Make Effective Use Of Continuous Integration



Setup a CI server (i.e. Jenkins, CircleCI, TravisCI, etc)

Research how to build using the SFDX CLI

Learn how to manage dependencies from one package to the downstream packages.

Trigger your builds from other build jobs of upstream dependencies

Setup two styles of build jobs

- CLI-based build job for each package
- Metadata-based build job for the remaining unpackaged code

Effective at revealing hidden “circular dependencies”

Breaking Down Standard and Custom SObjects

COMMON package has the minimal fields of the Account object.

-- FinancialForce

Account

-- Name

-- Owner

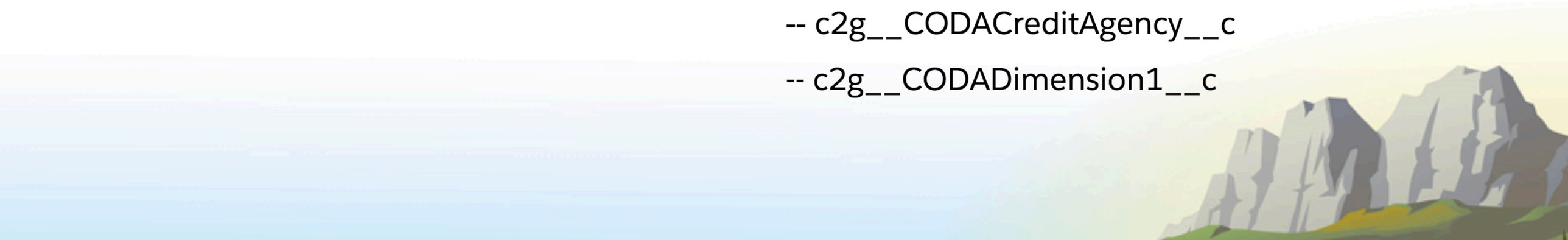
-- AnnualRevenue

-- Parent

-- Phone

-- c2g__CODACreditAgency__c

-- c2g__CODADimension1__c



Breaking Down Standard and Custom SObjects

COMMON package has the minimal fields of the Account object.

-- FinancialForce
-- Litify Matter Management

Your own unlocked packages will also add more fields

Account

-- Name
-- Owner
-- AnnualRevenue
-- Parent
-- Phone
-- c2g__CODACreditAgency__c
-- c2g__CODADimension1__c
-- litify_pm__SLA__c
-- litify_pm__Last_Called_at__c

Breaking Down Standard and Custom SObjects

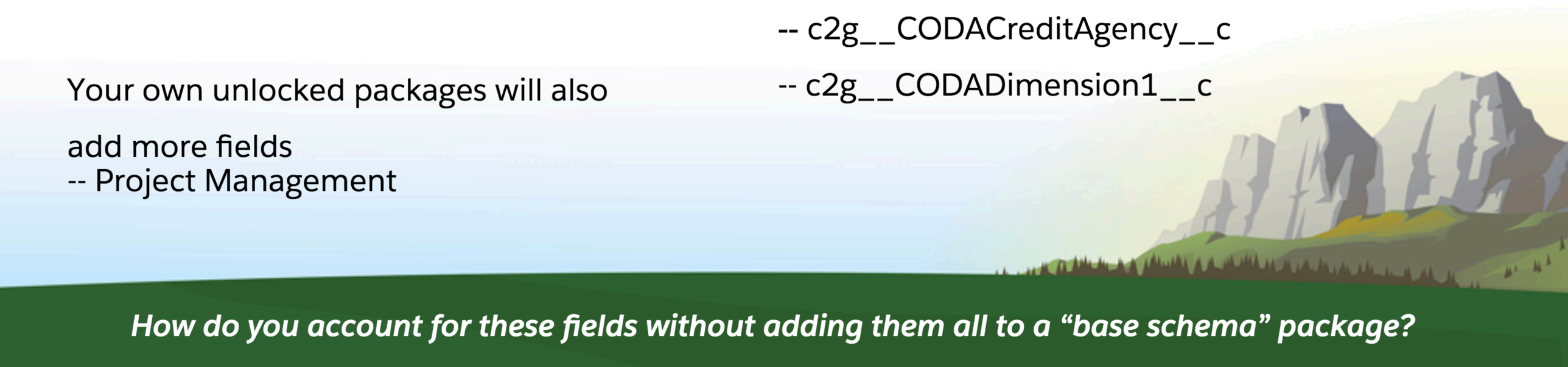
COMMON package has the minimal fields of the Account object.

-- FinancialForce
-- Litify Matter Management

Your own unlocked packages will also add more fields
-- Project Management

Account

-- Name
-- Owner
-- AnnualRevenue
-- Parent
-- Phone
-- c2g__CODACreditAgency__c
-- c2g__CODADimension1__c
-- litify_pm__SLA__c
-- litify_pm__Last_Called_at__c
-- Project_Create_Date__c



How do you account for these fields without adding them all to a "base schema" package?

Manage Object Field Additions Through Dependency Injection

Dependency injection of extra fields to selectors

COMMON

Apex Class
- common_AccountsSelector
  extends core_SObjectSelector

CORE

Apex Class
- core_SObjectSelector

Manage Object Field Additions Through Dependency Injection

Dependency injection of extra fields to selectors

COMMON

Apex Class
- common_AccountsSelector
  extends core_SObjectSelector

CORE

Apex Class
- core_SObjectSelector

Effective Query:

```
select id, name, AnnualRevenue
     , ParentId, Phone
  from Account
```

Manage Object Field Additions Through Dependency Injection

Dependency injection of extra fields to selectors

COMMON

Apex Class
- common_AccountsSelector
  extends core_SObjectSelector

CORE

Apex Class
- core_SObjectSelector

Effective Query:

```
select id, name, AnnualRevenue
     , ParentId, Phone
  from Account
```

PROJECT

Custom Field
- Account.Project_Create_Date__c

Manage Object Field Additions Through Dependency Injection

Dependency injection of extra fields to selectors

Effective Query:

```
select id, name, AnnualRevenue
     , ParentId, Phone
  from Account
```

COMMON

Apex Class
- common_AccountsSelector
  extends core_SObjectSelector

CORE

Apex Class
- core_SObjectSelector

Custom Metadata Type-based Dependency Injection

PROJECT

Custom Field
- Account.Project_Create_Date__c

Manage Object Field Additions Through Dependency Injection

Dependency injection of extra fields to selectors

Effective Query:

```
select id, name, AnnualRevenue
     , ParentId, Phone
  from Account
```

COMMON

Apex Class
- common_AccountsSelector
  extends core_SObjectSelector

CORE

Apex Class
- core_SObjectSelector

Custom Metadata Type
- core_SObjectFieldSetInclusion__mdt
  name = Field Set Name
  SObjectType__c = Sobject API name

Custom Metadata Type-based Dependency Injection

PROJECT

Custom Field
- Account.Project_Create_Date__c

Manage Object Field Additions Through Dependency Injection

Dependency injection of extra fields to selectors

Effective Query:

```
select id, name, AnnualRevenue
     , ParentId, Phone
  from Account
```

COMMON

Apex Class
- common_AccountsSelector
  extends core_SObjectSelector

CORE

Apex Class
- core_SObjectSelector

Custom Metadata Type
- core_SObjectFieldSetInclusion__mdt
  name = Field Set Name
  SObjectType__c = Sobject API name

Custom Metadata Type-based Dependency Injection

PROJECT

Custom Field
- Account.Project_Create_Date__c

FieldSet
- AccountRelatedFieldsFromProjects

Manage Object Field Additions Through Dependency Injection

Dependency injection of extra fields to selectors

Effective Query:

```
select id, name, AnnualRevenue
     , ParentId, Phone
  from Account
```

COMMON

Apex Class
- common_AccountsSelector
  extends core_SObjectSelector

CORE

Apex Class
- core_SObjectSelector

Custom Metadata Type
- core_SObjectFieldSetInclusion__mdt
  name = Field Set Name
  SObjectType__c = Sobject API name

Custom Metadata Type-based Dependency Injection

PROJECT

Custom Field
- Account.Project_Create_Date__c

FieldSet
- AccountRelatedFieldsFromProjects

Custom Metadata Record
- SObjectFieldSetInclusion.AccountFromProject

Manage Object Field Additions Through Dependency Injection

Dependency injection of extra fields to selectors

Effective Query:

```
select id, name, AnnualRevenue
     , ParentId, Phone

     , Project_Created_Date__c
  from Account
```

Custom Metadata Type-based Dependency Injection

COMMON

Apex Class
- common_AccountsSelector
  extends core_SObjectSelector

CORE

Apex Class
- core_SObjectSelector

Custom Metadata Type
- core_SObjectFieldSetInclusion__mdt
  name = Field Set Name
  SObjectType__c = Sobject API name

PROJECT

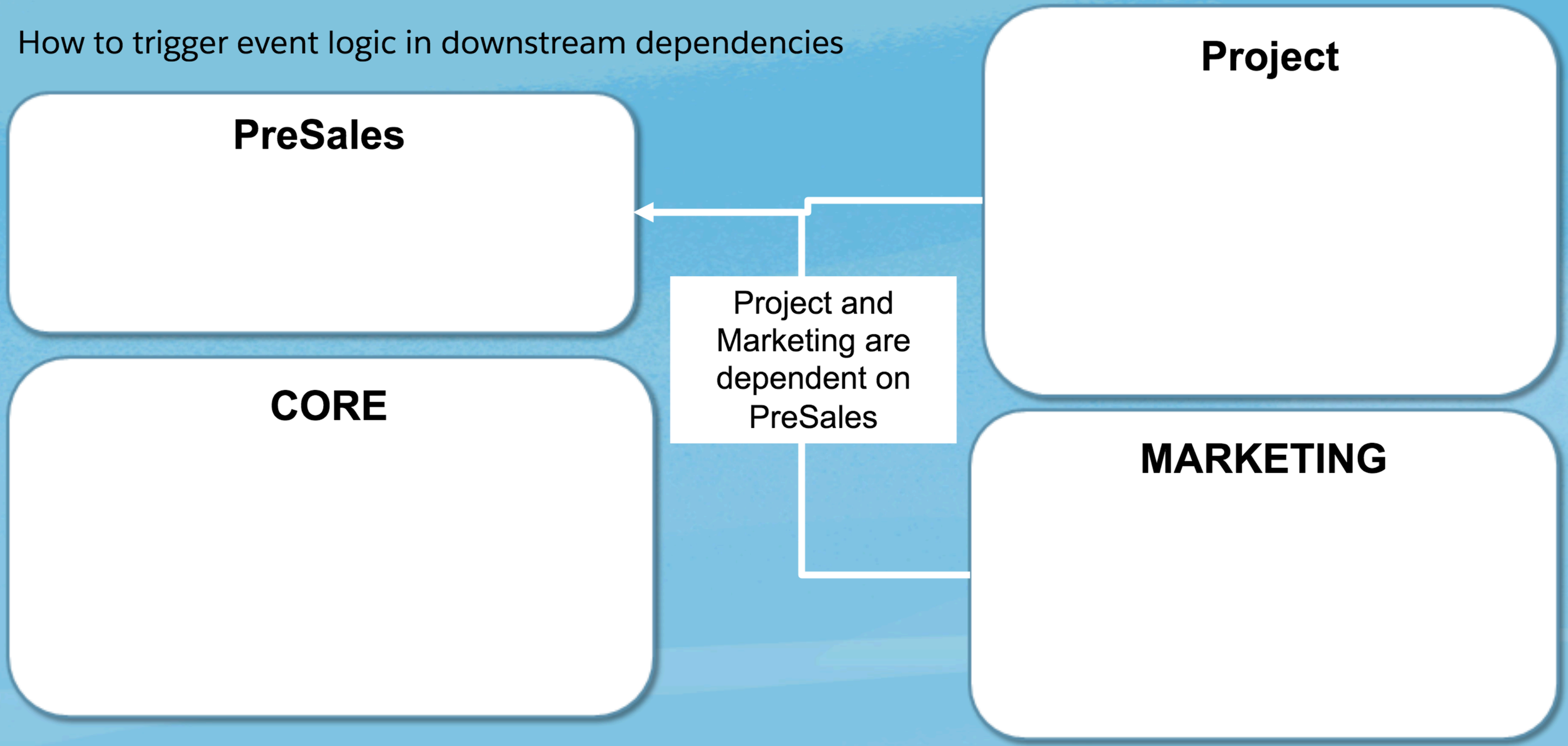
Custom Field
- Account.Project_Create_Date__c

FieldSet
- AccountRelatedFieldsFromProjects

Custom Metadata Record
- SObjectFieldSetInclusion.AccountFromProject

Inter-Package Communication

How to trigger event logic in downstream dependencies



Inter-Package Communication

How to trigger event logic in downstream dependencies

PreSales

Event

- Customer is retained and
  Pre-Sales record converts

CORE

Project

Response To Event

- New project record is created

MARKETING

Response To Event

- Logs conversion of presales to project data by campaign

Inter-Package Communication

How to trigger event logic in downstream dependencies

PreSales

Event

- Customer is retained and Pre-Sales record converts

CORE

Dependency Injection based Platform Event Subscriptions

Project

Response To Event

- New project record is created

MARKETING

Response To Event

- Logs conversion of presales to project data by campaign

Inter-Package Communication

How to trigger event logic in downstream dependencies

PreSales

Event
- Customer is retained and
  Pre-Sales record converts

CORE

Apex Interface
- core_IEventsConsumer

Custom Metadata Type
- core_EventConsumerSubscription__mdt

Apex Class
- core_PlatformEventsDistributor

Dependency Injection based Platform Event Subscriptions

Project

Response To Event
- New project record is created

MARKETING

Response To Event
- Logs conversion of presales to project data by campaign

Inter-Package Communication

How to trigger event logic in downstream dependencies

PreSales

Event
- Customer is retained and
  Pre-Sales record converts

CORE

Apex Interface
- core_IEventsConsumer

Custom Metadata Type
- core_EventConsumerSubscription__mdt

Apex Class
- core_PlatformEventsDistributor

Dependency Injection based Platform Event Subscriptions

Project

Apex Class
- project_EventConsumer
  extends core_IEventsConsumer

Custom Metadata Record
- Consume Presales Convert - Project

MARKETING

Apex Class
- marketing_EventConsumer
  extends core_IEventsConsumer

Custom Metadata Record
- Consume Presales Convert Marketing

Key Considerations

Adhere to the SOLID principles:
https://en.m.wikipedia.org/wiki/SOLID_(object-oriented_design)

- A package should be responsible for one thing only.
- It should be open for extension without needing to be changed directly.
- Logic should be able to be replaced if needed
- Use interfaces from a core package to define the structure of your packages
- Rely on interfaces from upstream packages – not the specific concreate implementations