1. 资产事件

如果你需要在资产创建、修改或删除时执行特定操作，可以创建一个系统来响应 AssetEvent 事件。

```
fn print_asset_event(mut asset_events:
EventReader<AssetEvent<Mesh>>) {
    for asset_event in asset_events.read() {
        match asset_event {
            AssetEvent::Added { id } => info!("asset event
added: {id}"),
            AssetEvent::Modified { id } => info!("asset
event modified: {id}"),
            AssetEvent::Removed { id } => info!("asset event
removed: {id}"),
            AssetEvent::Unused { id } => info!("asset event
unused: {id}"),
            AssetEvent::LoadedWithDependencies { id } =>
info!("asset event loaded: {id}"),
        }
    }
}
```

注意：如果你正在处理 Modified 事件并对数据进行可变访问，.get_mut 将会为同一个资产触发另一个 Modified 事件。如果不小心，这可能会导致无限循环！（由你自己的系统引起的事件）

2. 追踪加载进度

如果你想检查各种资产文件的状态，可以从 AssetServer 轮询。它会告诉你资产是否已加载、仍在加载、未加载或遇到错误。

要检查单个资产，你可以使用 asset_server.get_load_state(...)，并提供一个句柄或路径来引用该资产。

```
fn print_load_state(asset_server: Res<AssetServer>, suzanne:
Res<Suzanne>) {
    let Some(load_state) =
asset_server.get_load_state(suzanne.0.id()) else {
        return;
    };
    match load_state {
        LoadState::NotLoaded => info!("asset not loaded"),
        LoadState::Loading => info!("asset loading"),
        LoadState::Loaded => info!("asset loaded"),
        LoadState::Failed(error) => error!("asset failed:
{error}"),
    }
}
```

3. 访问资产数据

要从系统中访问实际的资产数据，请使用 Assets<T> 资源。

你可以使用句柄来识别所需的资产。

```
fn suzanne_to_cube(mut meshes: ResMut<Assets<Mesh>>,
suzanne: Res<Suzanne>) {
    let Some(mesh) = meshes.get_mut(suzanne.0.id()) else {
        return;
    };
    *mesh =
Cuboid::from_size(Vec3::splat(2.)).mesh().build();
}
```

4. 从代码创建资产

你也可以手动将资产添加到 Assets<T> 中。

为此，首先创建资产的数据（资产类型的实例），然后将其添加到 Assets<T> 资源中，以便 Bevy 存储和跟踪它。你将获得一个句柄来引用它，就像任何其他资产一样。

```
fn spawn_red_sphere(
    mut commands: Commands,
    mut meshes: ResMut<Assets<Mesh>>,
    mut materials: ResMut<Assets<StandardMaterial>>,
) {
    commands.spawn((
        MaterialMeshBundle {
            mesh: meshes.add(Sphere::new(1.)),
            material:
materials.add(Color::Srgba(Srgba::RED)),
            transform: Transform::from_xyz(-2.5, 0., 0.),
            ..default()
        },
        Name::new("Red Sphere"),
    ));
}
```