1. Commands （命令）

使用 Commands 从系统中生成/删除实体，添加/删除实体上的组件，管理资源。

```
fn change_player(mut commands: Commands, players:
Query<Entity, With<Player>>) {
    let mut players = players.iter();
    let player_1 = players.next().unwrap();
    // 从实体中删除组件，向实体插入组件

commands.entity(player_1).remove::<Player>().insert(Enemy);

    let player_2 = players.next().unwrap();
    // 保留实体中的部分组件
    commands.entity(player_2).retain::<Level>();

    let player_3 = players.next().unwrap();
    // 删除实体
    commands.entity(player_3).despawn();
}
```

2. 这些行为什么时候应用?

Commands 不会立即生效，因为当其他系统可以并行运行时，修改内存中的数据布局是不安全的。当你使用 Commands 执行的任何操作，都会排队等待稍后安全时应用。

在同一个 Schedule 中，你可以向系统添加 .before() / .after() 排序约束，Bevy 将自动确保在必要时应用 Commands ，以便第二个系统可以看到第一个系统中 Commands 应用后的情况。

```
.add_systems(
            Startup,
            (
                spawn_players,
                change_player,
            ).chain(),
        )
```

否则，命令通常在每个 Schedule 结束时被应用。处于不同 Schedule 的系统将会看到变化。例如，Startup 中生成的实体，可以在 Update 中看到。

```
.add_systems(Startup, spawn_players)
.add_systems(Update, complete_player_count)
```

3. 自定义 Commands

Commands 还可以作为一种便捷的方式来执行任何需要完全访问 ECS World 的自定义操作。你可以将任何自定义操作以延迟排队的方式运行，这与标准命令的工作方式相同。

```
fn custom_commands_spawn_player(mut commands: Commands) {
    commands.add(|world: &mut World| {
        world.spawn((Player, Level(45), Health(236)));
    });
}
```

4. 扩展 Commands API

如果你想要更集成的东西，就像 Bevy 的 Commands API 一样。你可以创建自定义类型并实现 Command Trait：

```
struct AddEnemyCommand {
    level: u32,
    health: u32,
}

impl Command for AddEnemyCommand {
    fn apply(self, world: &mut World) {
        world.spawn((
            Enemy,
            Level(self.level),
            Health(self.health),
        ));
    }
}

fn use_add_enemy_command(mut commands: Commands) {
    commands.add(AddEnemyCommand {
        level: 5,
        health: 28,
    });
}
```

如果你想让他更加好用，你可以创建一个扩展 Trait 来向 Commands 添加额外的方法：

```
trait AddEnemyCommandExt {
    fn add_enemy(&mut self, level: u32, health: u32);
}

impl<'w, 's> AddEnemyCommandExt for Commands<'w, 's> {
    fn add_enemy(&mut self, level: u32, health: u32) {
        self.add(AddEnemyCommand {
            level,
            health,
        });
    }
}

fn use_add_enemy_command_ext(mut commands: Commands) {
    commands.add_enemy(34, 356);
}
```