Upgradeable Contracts
Starknet separates contracts into classes and instances, making it simple to upgrade a contract's logic without affecting its state.
A contract class is the definition of the semantics of a contract. It includes the entire logic of a contract: the name of the entry points, the addresses of the storage variables, the events that can be emitted, etc. Each class is uniquely identified by its class hash. A class does not have its own storage: it's only a definition of logic.
Classes are typically identified by a [class hash][class hash doc]. When declaring a class, the network registers it and assigns a unique hash used to identify the class and deploy contract instances from it.
A contract instance is a deployed contract corresponding to a class, with its own storage.
Starknet natively supports upgradeable contracts through the `replace_class_syscall` [system call][syscalls doc], enabling simple contract upgrades without affecting the contract's state.
[class hash doc]: https://docs.starknet.io/documentation/architecture_and_concepts/Smart_Contracts/class-hash
[syscalls doc]: https://docs.starknet.io/documentation/architecture_and_concepts/Smart_Contracts/system-calls-cairo1/
Upgrading Contracts
To upgrade a contract, expose an entry point that executes `replace_class_syscall` with the new class hash as an argument:
```cairo,noplayground
{{#include ../listings/ch16-building-advanced-starknet-smart-contracts/listing_06_upgrade_with_syscall/src/lib.cairo}}
```
{{#label replace-class}}
<span class="caption">Listing {{#ref replace-class}}: Exposing `replace_class_syscall` to update the contract's class</span>
> Note: Thoroughly review changes and potential impacts before upgrading, as it's a delicate procedure with security implications. Don't allow arbitrary addresses to upgrade your contract.
Upgradeable Component
OpenZeppelin Contracts for Cairo provides the `Upgradeable` component that can be embedded into your contract to make it upgradeable. This component is a simple way to add upgradeability to your contract while relying on an audited library. It can be combined with the `Ownable` component to restrict the upgradeability to a single address, so that the contract owner has the exclusive right to upgrade the contract.
```cairo,noplayground
{{#include ../listings/ch16-building-advanced-starknet-smart-contracts/listing_07_oz_upgrade/src/lib.cairo}}
```
{{#label upgradeable-contract}}
<span class="caption">Listing {{#ref upgradeable-contract}} Integrating OpenZeppelin's Upgradeable component in a contract</span>
For more information, please refer to the [OpenZeppelin docs API reference][oz

upgradeability api].
[oz upgradeability api]: https://docs.openzeppelin.com/contracts-cairo/0.9.0/api/upgrades