Most of the out-of-the-box configurations are probably fine for ArchivesSpace so I would like to make the fewest number of changes to the configuration as possible. Some changes that are currently in GitHub are:

1. Do not run the code on the build and docs directories
2. For arrays, instead of using %i and %w, use brackets (Style/SymbolArray and Style/WordArray)
3. Disabling Style/Documentation which requires top-level documentation of classes and modules

There are a few out-of-the-box configurations I would like to talk about as a group. For the ones without a bolded comment, I could go either way on but think someone in the group might have an opinion.

Currently disabled, that maybe we want to enable:

1. Layout/ClassStructure: Enforces a configured order of definitions within a class body.
2. Lint/NumberConversion: Checks unsafe usage of number conversion methods. **Not sure if this really matters for ArchivesSpace or not.**
3. Style/ImplicitRuntimeError: Use `raise` or `fail` with an explicit exception class and message, rather than just a message.
4. Style/StringHashKeys: Prefer symbols instead of strings as hash keys. **Definitely inconsistent across the code base**

Currently enabled, that maybe we want to disable:

1. Layout/SpaceAroundEqualsInParameterDefault: Enforces that the equals signs in parameter default assignments have surrounding space.
2. Lint/BooleanSymbol: Check for `:true` and `:false` symbols. **I am concerned they aren't consistently used as a symbol in ArchivesSpace code.**
3. Lint/RequireParentheses: Use parentheses in the method call to avoid confusion about precedence.
4. Lint/ToJSON: Ensure #to\_json includes an optional argument
5. Lint/UnifiedInteger: Use Integer instead of Fixnum or Bignum.
6. Lint/UriEscapeUnescape: `URI.escape` method is obsolete and should not be used. Instead, use `CGI.unescape`, `URI.decode\_www\_form` or `URI.decode\_www\_form\_component` depending on your specific use case." **Need to determine appropriate replacement.**
7. Style/BlockComments: Do not use block comments.
8. Style/BracesAroundHashParameters: Enforce no braces around hash parameters. **I think braces make it more obvious that is it a hash so it should enforce braces**
9. Style/ClassVars: Avoid the use of class variables.
10. Style/ColonMethodCall: Do not use :: for method call. Instead use '.'
11. Style/ColonMethodDefinition: Do not use :: for defining class methods.
12. Style/EmptyLiteral: Prefer literals to Array.new/Hash.new/String.new.
13. Style/GlobalVars: Do not introduce global variables.

Currently enabled, for which we may want to increase the maximum or disable altogether:

1. Metrics/AbcSize: A calculated magnitude based on number of assignments, branches, and conditions. Maximum is 15
2. Metrics/BlockLength: Avoid long blocks with many lines. Maximum lines is 25
3. Metrics/BlockNesting: Avoid excessive block nesting. Maximum is 3
4. Metrics/ClassLength: Avoid classes longer than 100 lines of code. Maximum is 100
5. Metrics/CyclomaticComplexity: A complexity metric that is strongly correlated to the number of test cases needed to validate a method. Maximum is 6
6. Metrics/LineLength: Limit lines to 80 characters. Maximum is 80
7. Metrics/MethodLength: Avoid methods longer than 10 lines of code. Maximum is 10
8. Metrics/ModuleLength: Avoid modules longer than 100 lines of code. Maximum is 100 and ignore comments
9. Metrics/ParameterLists: Avoid parameter lists longer than three or four parameters. Maximum is 5
10. Metrics/PerceivedComplexity: A complexity metric geared towards measuring complexity for a human reader. Maximum is 7